



## Instruction Manual

Version 1.2

8/7/2014

Justin Blaber (jblaber3@gatech.edu)

Antonia Antoniou (antonia.antoniou@me.gatech.edu)

Georgia Institute of Technology

## Table of Contents

### 1 - Installation

- [1.1 - Installation Requirements](#)
- [1.2 - MEX Setup](#)
- [1.3 - Automatic Installation](#)
- [1.4 - Manual Installation](#)

### 2 - User Guide – General DIC

- [2.1 - Program Flow](#)
- [2.2 - Setting the Reference Image](#)
- [2.3 - Setting the Current Image\(s\)](#)
- [2.4 - Setting the Region of Interest](#)
- [2.5 - Setting DIC Parameters](#)
- [2.6 - DIC Analysis](#)
- [2.7 - Formatting Displacements](#)
- [2.8 - Strain Analysis](#)
- [2.9 - Plotting](#)
- [2.10 - Directly Obtaining and Altering DIC Data](#)

### 3 - User Guide – High Strain

- [3.1 - High Strain DIC Analysis](#)

### 4 - User Guide – Discontinuous Displacements

- [4.1 - Discontinuous Displacements DIC Analysis](#)

## 1.1 - Installation Requirements

### Version Requirements:

- *Required:* R2009a+
- *Recommended:* R2012a+

**NOTE:** Ncorr was developed on MATLAB R2009a and has not been tested on prior versions. It works best on MATLAB R2012a and above, as R2009a has some problems with the stacking order of objects in the plotting tool.

### Toolbox Requirements:

- *Required:* Image Processing Toolbox
- *Required:* Statistics Toolbox

### Operating System Requirements:

- *Required:* Windows or Linux
- *Recommended:* Windows

**NOTE:** Ncorr was developed on Windows 7. It has also been tested on Linux Ubuntu 11.10. Ncorr currently will not work on Mac OS.

### MEX Compiler Requirements:

- *Recommended:* Visual Studio 2005 (8.0) + or GCC 4.2+

**NOTE:** Ncorr utilizes the STL library. It also has an option to use OpenMP for users with multicore processors. OpenMP requires the compilers listed above; if the use of OpenMP is omitted, then any C++ compiler that supports the STL library should work.

## 1.2 - MEX Setup

First, make sure MEX has been set up properly in MATLAB. Type "mex -v" in the MATLAB terminal as shown below:

```
EDU>> mex -v
```

The output will either be:

```
Select a compiler:
[1] Lcc-win32 C 2.4.1 in C:\PROGRA~2\MATLAB\R2009A~1\sys\lcc
[2] Microsoft Visual C++ 2008 Express in C:\Program Files (x86)\Microsoft Visual Studio 9.0
```

or:

```
-> Default options filename found in C:\Users\.....\AppData\Roam
-----
-> Options file      = C:\Users\.....\AppData\Roaming\Ma
    MATLAB            = C:\PROGRA~2\MATLAB\R2009A~1
-> COMPILER          = cl
-> Compiler flags:
    COMPFLAGS        = /c /Zp8 /GR /W3 /EHs /D_CRT_SECURE_NO_
    OPTIMFLAGS       = /O2 /Oy- /DNDEBUG
    DEBUGFLAGS       = /Z7
    arguments         =
    Name switch       = /Fo
-> Pre-linking commands =
-> LINKER           = link
-> Link directives:
    LINKFLAGS        = /dll /export:mexFunction /LIBPATH:"C:
    * MEXDEPENDENCIES.lcc
    - /NODEFAULTLIB:MSVCR90.dll"
```

In the first case, MEX has not been set up so, so choose Microsoft Visual C++ (or GCC on Linux systems) and install it. In the second case, MEX has already been set up, but check to make sure the correct compiler is installed. If "**COMPILER = cl**" or "**COMPILER = gcc**," then MEX is set up with the **correct** compiler. If a different compiler is installed, rerun the MEX setup by typing "mex -setup" in the MATLAB terminal as shown below:

```
>> mex -setup
```

And select either Microsoft Visual C++ (or GCC on Linux systems) from the list that appears.

### 1.3 - Automatic Installation

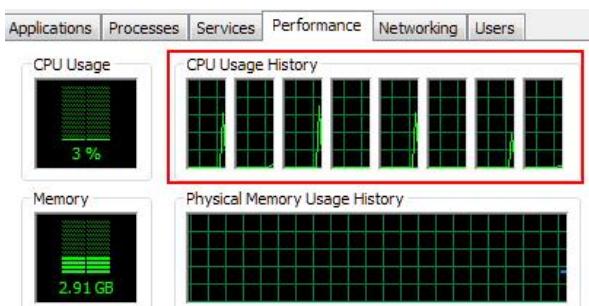
After MEX is set up correctly, get the latest version of Ncorr from the ncorr.com website *downloads* section (as of this manual, the latest version is v1.2), as shown below:

The screenshot shows the Ncorr v1.2 website. At the top, there's a banner with text: "Open source 2D-DIC MATLAB software", "RGDIC - Contiguous Circular Subsets - Biquintic B-Spline Interpolation - High Strain Analysis", and "Inverse Compositional Method Image Alignment - Mex - OpenMP". Below the banner is a navigation bar with links: HOME, DOWNLOADS, APPLICATIONS, DIC ALGORITHMS, and DATA COLLECTION. The DOWNLOADS link is highlighted. Under the DOWNLOADS link, there are two sections: "Program" and "Downloads". The "Program" section contains a link labeled "ncorr\_v1.2 - zip" which is highlighted with a red box. The "Downloads" section contains a note: "Click on the link under "Program" on the menu to the left to download the zip file containing .m, .cpp, and .h source files as well as the mex files that automatically compiles the MEX files before running them."

Then, navigate to the directory where you saved Ncorr. Make sure none of the files have been moved or altered. From there, type "handles\_ncorr = ncorr" in the MATLAB terminal as shown below:

The screenshot shows a MATLAB environment. On the left is the "Current Directory" browser showing a folder named "ncorr\_matlab" containing numerous files. On the right is the "Command Window" with the command "EDU>> handles\_ncorr = ncorr;" entered. A small window titled "Student Version : Openmp..." is open, asking if "Openmp support" is required. It includes checkboxes for "Cores:" (set to 1) and "Manually specify GNU compiler (if installed with GNUMEX)".

A dialogue box will appear about OpenMP support. Click the checkbox for OpenMP if you have a multicore processor and want multicore support. The number of cores on your system is most easily determined by checking the task manager on Windows (or system processes on Linux) as shown below:



In my case, my computer has 8 logical cores that can be utilized for computation.

**NOTE:** If you use all your cores during computation, generally any other applications on the computer will slow down drastically (such as browsing the internet), so if you plan on using your computer during computation, then it might be best to leave out a core or two.

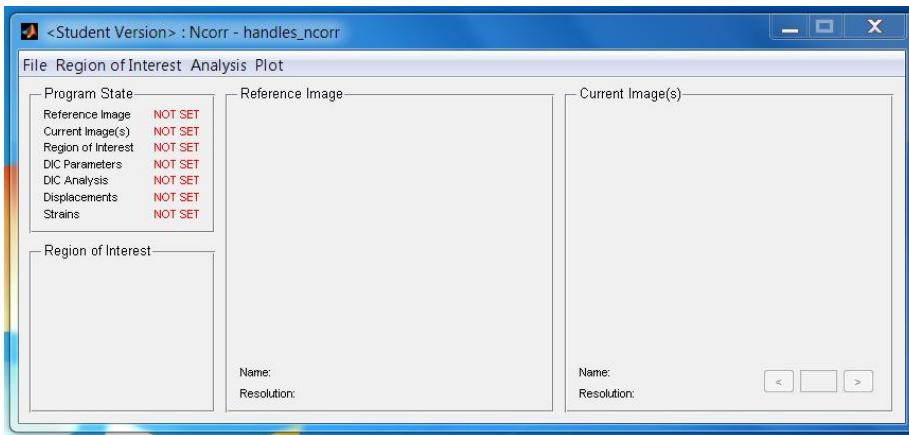
The manual compiler specification option is there for the case that GCC is installed on Windows through gnumex. This is probably won't be the case for most users, so when in doubt leave it unchecked.

During installation, a pop-up for setting the path will appear. The path needs to be set in order to upload images from other directories. This is only a temporary solution and the dialogue box will reappear if MATLAB is closed and reopened. For a more permanent solution, you can set the path manually through: *File > Set Path > Add Folder* from the MATLAB GUI.

If the installation proceeds as normal, the following should appear in the terminal:

```
Installing standard_datatypes... Please wait
Installing ncorr_datatypes... Please wait
Installing ncorr_lib... Please wait
Installing ncorr_alg_formmask... Please wait
Installing ncorr_alg_formregions... Please wait
Installing ncorr_alg_formboundary... Please wait
Installing ncorr_alg_formthreaddiagram... Please wait
Installing ncorr_alg_formunion... Please wait
Installing ncorr_alg_extrapdata... Please wait
Installing ncorr_alg_adddisp... Please wait
Installing ncorr_alg_convert... Please wait
Installing ncorr_alg_disograd... Please wait
Installing ncorr_alg_calcseeds... Please wait
Installing ncorr_alg_rgdic... Please wait
```

And then a GUI for Ncorr should appear:



If this is the case, Ncorr has most likely installed correctly and the installation should be complete. The next time you open MATLAB, you can open Ncorr again by typing "handles\_ncorr = ncorr" into the MATLAB terminal and the GUI should appear without having to repeat the installation process.

If compilation fails, then the following error message will appear:



If all the above steps have been followed correctly, then first try to install Ncorr without OpenMP. If compilation still fails, then you may need to complete the installation manually as described in the next section. If the installation proceeded correctly, then you can skip the rest of the installation section and move onto the user guides.

## 1.4 - Manual Installation

These steps show how to perform the manual installation of Ncorr, which involves the compilation of MEX files as well as the creation of an "ncorr\_installinfo.txt" file which contains information about multicore support.

The first step is to comment out the automatic compile section in the **ncorr.m** file. Add "%{" before the compile section and then "%}" after it as shown below:

```

C:\Users\jucestain\Desktop\ncorr_matlab\ncorr.m
File Edit Text Go Cell Tools Debug Desktop Window Help
File Edit Text Go Cell Tools Debug Desktop Window Help Stack: Base fx
2636 -
2637 -
2638 -
2639 -    end
2640 %(
2641 % BEGIN COMPILE SECTION
2642 % COMMENT THIS SECTION OUT AND MANUALLY COMPILE IF
2643 % COMPILED FAILS!!! THIS SECTION WILL ONLY SUPPORT OPENMP
2644 % FOR COMPILERS (CL AND GCC) WHICH WERE INSTALLED THE
2645 % STANDARD WAY (mex -setup) AND THROUGH GNUMEX. IF THE
2646 % COMPILER WAS MANUALLY INSTALLED A DIFFERENT WAY, OPENMP
2647 % WILL PROBABLY NOT COMPILE CORRECTLY, ALTHOUGH SINGLE
2648 % THREADED INSTALLATION SHOULD BE FINE. INSTRUCTIONS FOR
2649 %
2650 %-----%
2651 %-----%
2652 %-----%
2653 %-----%
2654 %-----%
2655 %-----%
2656 %-----%
2657 %-----%
2658 %-----%
2659 %-----%
2660 %-----%
2661 %-----%
2662 %-----%
2663 %-----%
2664 %-----%
2665 %-----%
2666 %-----%
2667 %-----%
2668 %-----%
2669 %-----%
2670 %-----%
2671 %-----%
2672 %-----%
2673 %-----%
2674 %-----%
2675 %-----%
2676 %-----%
2677 %-----%
2678 %-----%
2679 %-----%
2680 %-----%
2681 %-----%
2682 %-----%
2683 %-----%
2684 %-----%
2685 %-----%
2686 %-----%
2687 %-----%
2688 %-----%
2689 %-----%
2690 %-----%
2691 %-----%
2692 %-----%
2693 %-----%
2694 %-----%
2695 %-----%
2696 %-----%
2697 %-----%
2698 %-----%
2699 %-----%
2700 %-----%
2701 %-----%
2702 %-----%
2703 %-----%
2704 %-----%
2705 %-----%
2706 %-----%
2707 %-----%
2708 %-----%
2709 %-----%
2710 %-----%
2711 %-----%
2712 %-----%
2713 %-----%
2714 %-----%
2715 %-----%
2716 %-----%
2717 %-----%
2718 %-----%
2719 %-----%
2720 %-----%
2721 %-----%
2722 %-----%
2723 %-----%
2724 %-----%
2725 %-----%
2726 %-----%
2727 %-----%
2728 %-----%
2729 %-----%
2730 %-----%
2731 %-----%
2732 %-----%
2733 %-----%
2734 %-----%
2735 %-----%
2736 %-----%
2737 %-----%
2738 %-----%
2739 %-----%
2740 %-----%
2741 %-----%
2742 %-----%
2743 %-----%
2744 %-----%
2745 %-----%
2746 %-----%
2747 %-----%
2748 %-----%
2749 %-----%
2750 %-----%
2751 %-----%
2752 %-----%
2753 %-----%
2754 %-----%
2755 %-----%
2756 %-----%
2757 %-----%
2758 %-----%
2759 %-----%
2760 %-----%
2761 %-----%
2762 %-----%
2763 %-----%
2764 %-----%
2765 %-----%
2766 %-----%
2767 %-----%
2768 %-----%
2769 %-----%
2770 %-----%
2771 %-----%
2772 %-----%
2773 %-----%
2774 %-----%
2775 %-----%
2776 %-----%
2777 %-----%
2778 %-----%
2779 %-----%
2780 %-----%
2781 %-----%
2782 %-----%
2783 %-----%
2784 %-----%
2785 %-----%
2786 %-----%
2787 %-----%
2788 %-----%
2789 %-----%
2790 %-----%
2791 %-----%
2792 %-----%
2793 %-----%
2794 %-----%
2795 %-----%
2796 %-----%
2797 %-----%
2798 %-----%
2799 %-----%
2800 %-----%
2801 %-----%
2802 %-----%
2803 %-----%
2804 %-----%
2805 %-----%
2806 %-----%
2807 %-----%
2808 %-----%
2809 %-----%
2810 %-----%
2811 %-----%
2812 %-----%
2813 %)
2814 %-----%
2815 %-----%
2816 end

```

The next step is to manually compile all the necessary libraries and MEX files. At the time of writing this manual, there are three basic library .cpp files, and eleven MEX files which are listed in the table below:

Library Files	standard_datatypes.cpp ncorr_datatypes.cpp ncorr_lib.cpp
MEX Files	ncorr_alg_formmask.cpp ncorr_alg_formregions.cpp ncorr_alg_formboundary.cpp ncorr_alg_formthreaddiagram.cpp ncorr_alg_formunion.cpp ncorr_alg_extrapdata.cpp ncorr_alg_adddisp.cpp ncorr_alg_convert.cpp ncorr_alg_dispgrad.cpp ncorr_alg_calcseeds.cpp ncorr_alg_rgdc.cpp

Start by compiling the libraries "standard\_datatypes.cpp," "ncorr\_datatypes.cpp," and "ncorr\_lib.cpp." as object files by using the "-c" flag with MEX as shown below:

```
EDU>> mex -c standard_datatypes.cpp
EDU>> mex -c ncorr_datatypes.cpp
EDU>> mex -c ncorr_lib.cpp
```

Next, compile the following MEX files with:

```
EDU>> mex ncorr_alg_formmask.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_formregions.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_formboundary.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_formthreaddiagram.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_formunion.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_extrapdata.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_adddisp.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_convert.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_disprgrad.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
```

**NOTE:** When compiling the MEX files with linux, the standard\_datatypes and ncorr\_datatypes libraries will be compiled with an ".o" extension instead of an ".obj" extension.

The last two files that need to be compiled are "ncorr\_alg\_calcseeds.cpp" and "ncorr\_alg\_rgdic.cpp." These files utilize OpenMP and thus needs to be compiled with certain compiler flags. If multithreaded support is not desired, then the files can be compiled like the other MEX files in the following way:

```
EDU>> mex ncorr_alg_calcseeds.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
EDU>> mex ncorr_alg_rgdic.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj
```

Otherwise, flags need to be passed to the compiler. The required flags are shown in the table below:

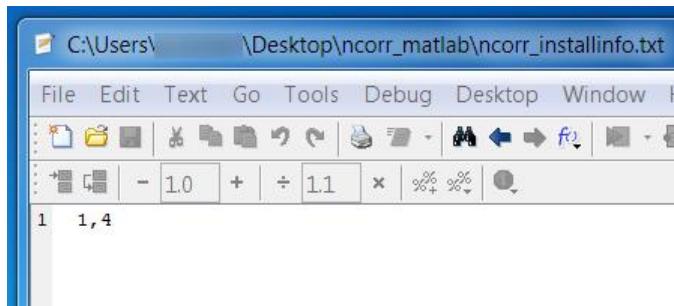
OS/Compiler	Flags
Windows with Visual Studio	COMPFLAGS="\$COMPFLAGS /openmp /DNCORR_OPENMP"
Windows with GCC (gnumex)	COMPFLAGS="\$COMPFLAGS -fopenmp -DNCORR_OPENMP" GM_ADD_LIBS="\$GM_ADD_LIBS -lgomp"
Linux with GCC	CXXFLAGS="\\$CXXFLAGS -fopenmp -DNCORR_OPENMP" CXXLIBS="\\$CXXLIBS -lgomp"

An example compilation for these files with the appropriate flags is shown below for a system with Windows and Visual studio:

```
EDU>> mex ncorr_alg_calcseeds.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj COMPFLAGS="$COMPFLAGS /openmp /DNCORR_OPENMP"
EDU>> mex ncorr_alg_rgdic.cpp standard_datatypes.obj ncorr_datatypes.obj ncorr_lib.obj COMPFLAGS="$COMPFLAGS /openmp /DNCORR_OPENMP"
```

The important thing to note here is to make sure you use the correct name and format for the compiler flags, which depend on **both the operating system and the compiler**. If compilation still fails or more assistance is needed then please email the author of Ncorr at jblaber3@gatech.edu.

At this point, all the MEX files should be compiled correctly. The next step is to create a file called "ncorr\_installinfo.txt" and then fill it with the following:



The first number represents whether or not OpenMP support exists, and should be either 0 or 1 (0 = no OpenMP support; 1 = OpenMP support). The second number represents how many threads you want the DIC analysis to run on (this should be a number greater than or equal to 1).

**NOTE:** Make sure to separate the numbers by a comma; the format should be: "#,#".

After the above is complete, type "handles\_ncorr = ncorr" into the MATLAB terminal to bring the GUI up. If no error messages appear, then the installation should be complete. As a double check, make sure the options you specified in the "ncorr\_installinfo.txt" file were loaded correctly by typing "handles\_ncorr" in the MATLAB terminal to view its properties:

```
EDU>> handles_ncorr = ncorr
handles_ncorr =
    ncorr handle
Properties:
    handles_gui: [1x1 struct]
    reference: [0x0 struct]
    current: [0x0 struct]
    data_dic: [1x1 struct]
    support_openmp: 1
    total_cores: 4
Methods, Events, Superclasses
```

Lastly, I suggest following the user guide and trying all Ncorr options with example images to ensure everything is working properly.

## 2.1 – Program Flow

For the general DIC section, I am using the “plate hole” sample from SEM’s DIC challenge. The formatted images, along with a ROI, are available off the Ncorr website if the user would like to follow along:

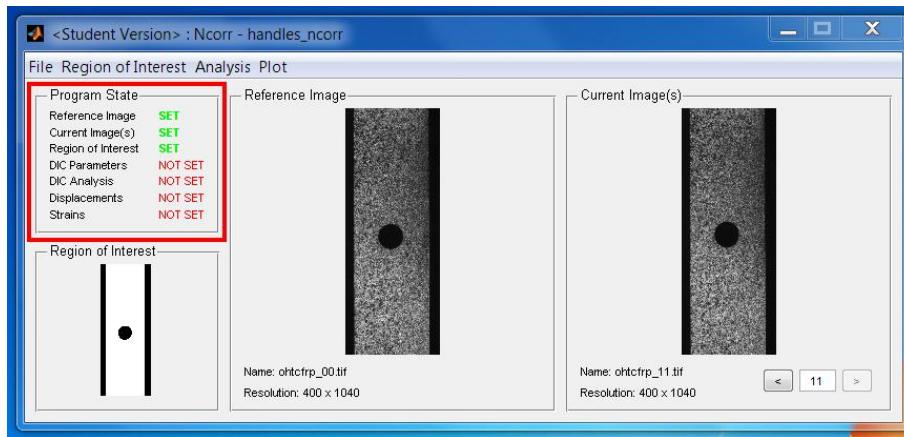
<http://ncorr.com/download/sample12.zip>

At this point, it is assumed Ncorr has been successfully compiled and installed on the user's computer.

The work flow of Ncorr is as follows:

1. Set Reference Image
2. Set Current Image(s)
3. Set Region of Interest (ROI)
  - *Dependencies:* Requires reference image or current image(s) to be set first
4. Set DIC Parameters
  - *Dependencies:* Requires reference image, current image(s), and ROI to be set first
5. DIC Analysis
  - *Dependencies:* Requires DIC parameters to be set first
6. Format Displacements
  - *Dependencies:* Requires DIC Analysis to be run first
7. Calculate Strains
  - *Dependencies:* Requires displacements to be formatted first.

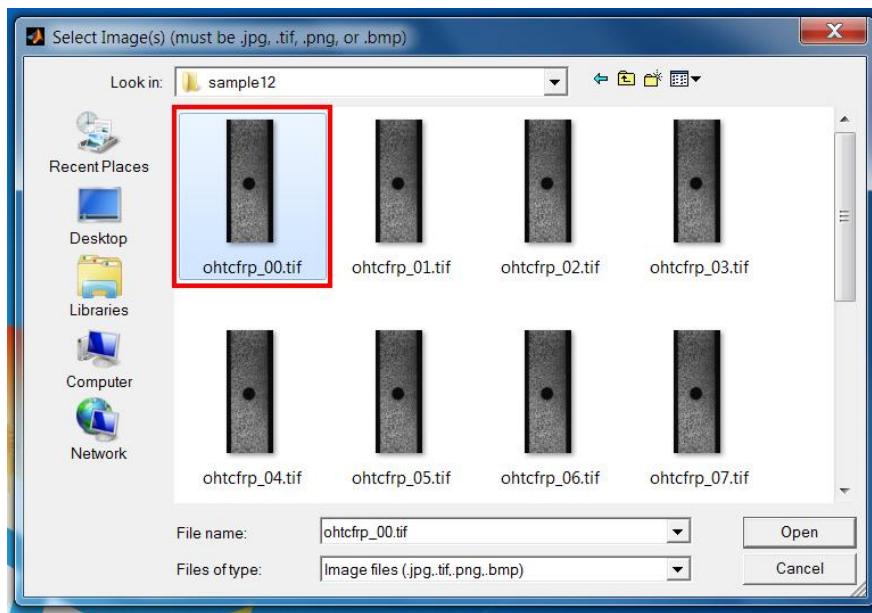
Any of these steps can be altered at any stage, so long as the dependencies have been met. The state of the program is visible at the top left corner of the main Ncorr GUI as shown below:



This lets the user know which stage they are currently on and which should come next. The state is listed in sequential order for convenience. In the example above, the state lets the user know that the reference image, current image, and ROI are loaded and that setting the DIC parameters needs to proceed next.

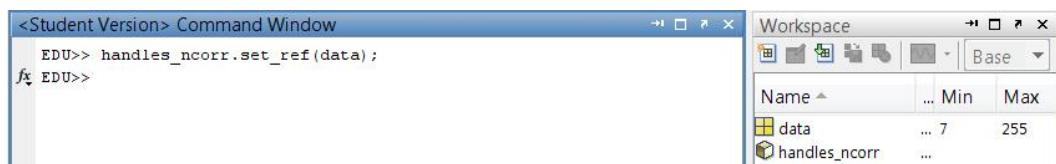
## 2.2 - Setting the Reference Image

There are two ways to set the reference image. The first way is to use the Ncorr GUI by going to *File > Load Reference Image* which results in the following:



**NOTE:** The reference image means the initial or first image you take of the sample, before any deformation occurs.

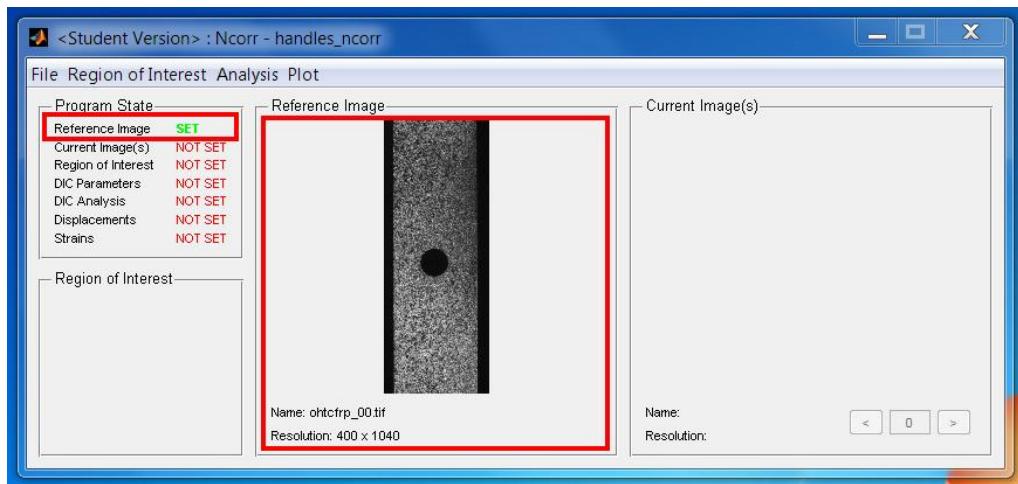
The other way to load a reference image is to do it through the MATLAB terminal. You can set the reference image by typing "handles\_ncorr.set\_ref(data)," where "data" is a 2D matrix (of type double, uint16, or uint8) containing image grayscale values, as shown below:



The reason for this feature is because sometimes the reference image is obtained through a series of images which are averaged together or processed in some way. If this is the case, the reference image will be stored as an array in the base workspace, and it's much more convenient to load it directly than to save it as an image, which can also lead to loss of data through image compression or binning of gray scale values.

Lastly, when the reference image is loaded it should appear as shown below:

## Setting the Reference Image - 2.2



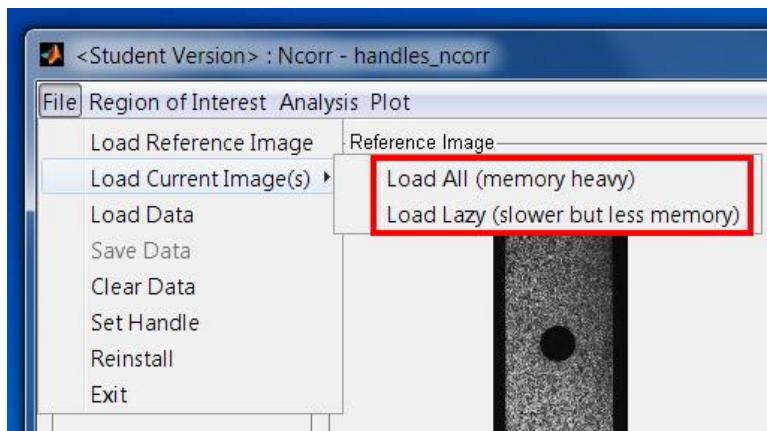
## 2.3 - Setting the Current Image(s)

First of all, one of the main differences between setting the current image(s) vs the reference image is that more than one image may be loaded for the current (deformed) configuration. Because of this, the program needs a way to order the current image(s) if more than one is loaded. To accomplish this, the images must have the name format shown below:

name\_#.ext

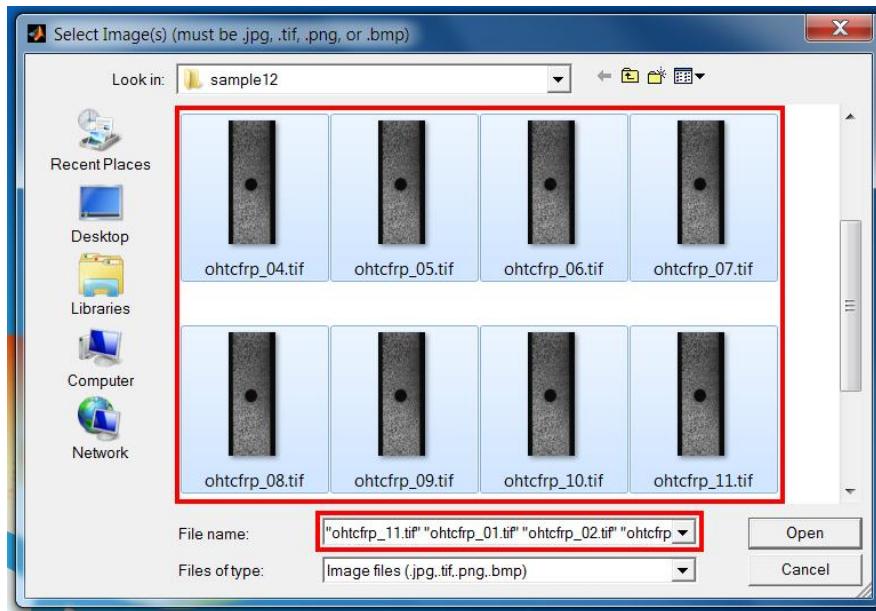
Where "name" is the name of the image set, "#" is a number associated with the image, and "ext" is the image extension, which must either be .jpg, .tif, .png, or .bmp (an example would be "sample\_15.png"). Note that this naming convention is only required if more than one current image is loaded through the GUI.

Setting the current image(s) through the GUI also differs from the reference image, in that there are two different loading schemes, "Load All" and "Load Lazy," which are shown by going to *File > Load Current Image(s)*.



"Load all" will store all the images into the workspace. If you are analyzing a large number of images, this will be problematic since it may result in running out of RAM space. However, when using lazy loading, only the name and path of the images are stored; the images are loaded based on their stored location "on demand." This means that, when doing analysis in Ncorr using lazy loaded images, the user must make sure not to move the images into a different directory or alter them. Furthermore, since they are loaded on demand and then discarded afterward, this means some of the analyses may proceed a little slower. But, from my experience, the difference is not that noticeable. When in doubt, use lazy loading.

When loading through the GUI (for both "load all" and "lazy load"), the following window should appear:



**NOTE:** Take note of the naming convention above. All of the current images follow the "name\_.#.ext" format, where name = "ohtcfpr" and ext = "tif".

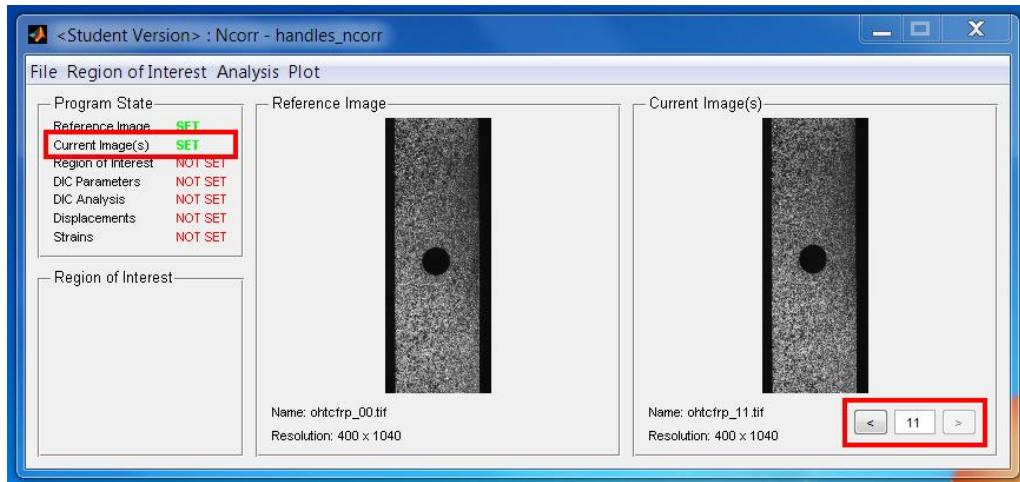
Simply drag and select all the current images you want to process. For this example, the images with a post-fix of "01" to "11" are selected).

Similar to the reference image, there is also an option to set the current image(s) directly through the Ncorr handle. To do this, you can type "handles\_ncorr.set\_cur(data)" through the MATLAB terminal, as shown below:



In the figure above, only a single current image is loaded, so it exists as a grayscale array of type double, uint16, or uint8. To load more than one image, each current image gray scale array must be stored together in a *cell array*. Furthermore, the ordering specified in the cell array is the order that Ncorr uses to sort the images.

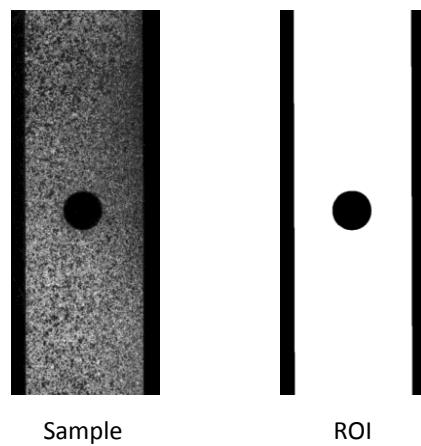
After the current images are uploaded, there are buttons on the bottom right of the Ncorr GUI which become enabled and allow you to scroll through the images as shown below:



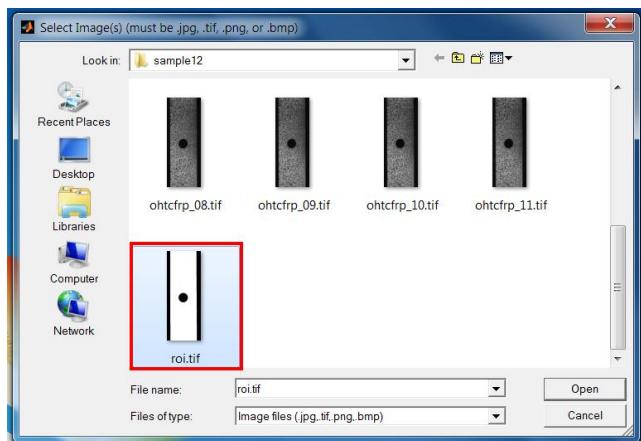
The current images should be ordered properly according to their number, and the last image in the sequence will appear initially by default. It's a good check to scroll through them to ensure the order is correct.

## 2.4 - Setting the Region of Interest

There are two ways to set the region of interest (ROI): through the GUI or through the MATLAB terminal. The most important thing to note here is that the ROI should be an array of the same size as the reference image or current image (this depends on which type of analysis is done, which is explained in greater detail in later sections. For this example, the ROI is set with respect to the *reference image*). Furthermore, white regions represent the ROI, and black regions are outside the ROI. It is also recommended to leave a black border near the edges of the image, although the ROI used in this example doesn't, in order to illustrate a couple points.

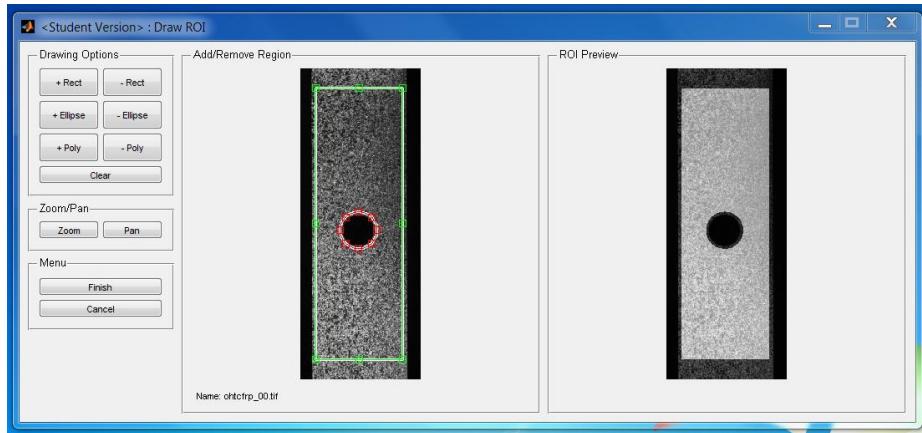


**Load ROI:** The first way to set the ROI is to load it from an image by going to: *Region of Interest > Set Reference ROI* and then pressing the *Load ROI* button. This is the recommended way to set a ROI because you can use a program like Photoshop to accurately trace an outline of the sample in order to obtain the best ROI. The only problem with this method is that it can be tedious. Anyway, a GUI should appear as shown below:



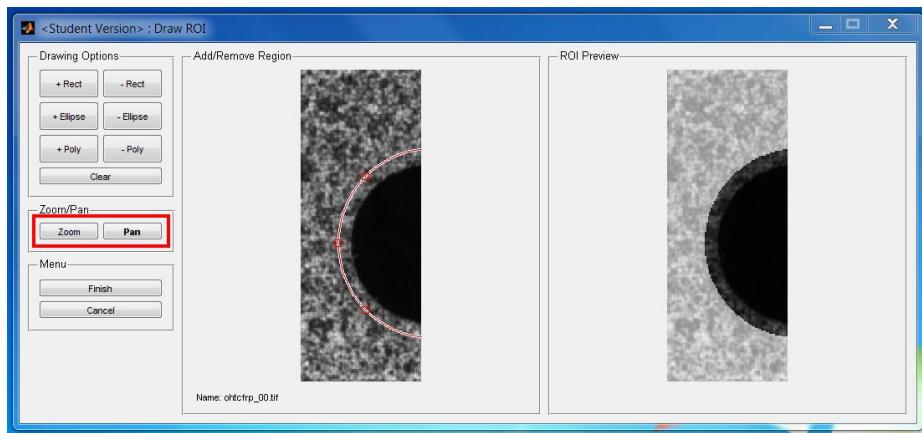
For the example used in this user guide, the ROI was set using this method.

**Draw ROI:** The second way to set the ROI is to draw the ROI directly in MATLAB by using *Region of Interest > Set Reference ROI* and then pressing the *Draw ROI* button. This is the preferred method for preliminary analysis because it can be done quickly. An example ROI is shown below and was constructed using "+ Poly" followed by "- Ellipse," where the "+" prefix indicates adding portions to the ROI; "-" subtracts regions:



**NOTE:** The green shapes represent regions which add to the ROI, red shapes represent regions that subtract from the ROI, and blue shapes represent regions which are active (i.e clicked on)

To obtain better precision when drawing ROIs, you can zoom and pan the view. Zooming can be enabled by clicking the zoom button once and disabled by clicking it again (or by clicking a different button, such as adding a region or pressing the pan button). An example is shown below:



It is also possible to delete specific shapes by right clicking them and selecting "Delete," as shown below:



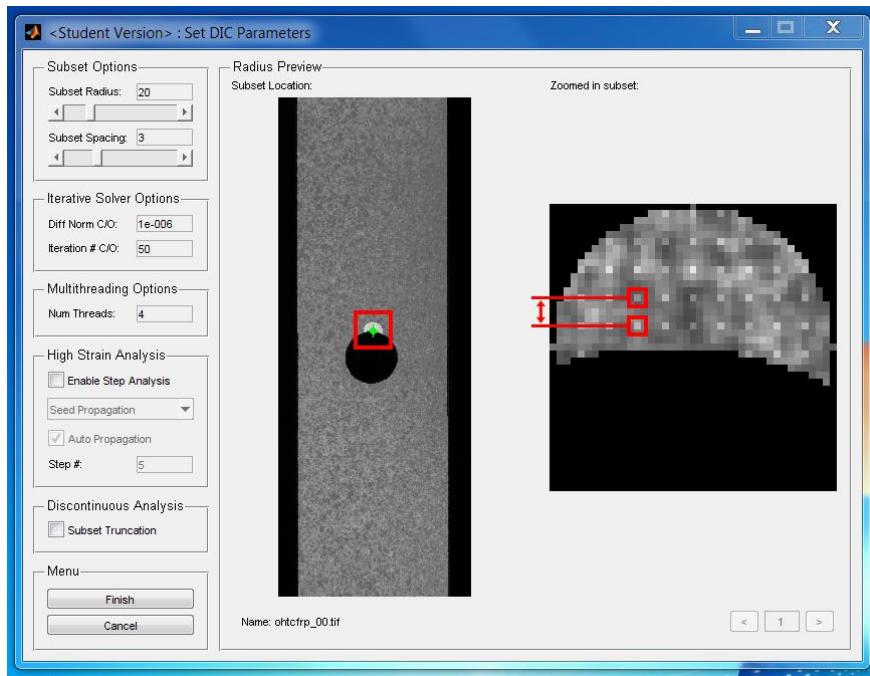
**MATLAB Terminal:** Once again, similar to loading the reference and current image through the MATLAB terminal, typing "handles\_ncorr.set\_roi\_ref(data)" will set the ROI. Note that data must be of type logical and be the same size as the reference image. An example is shown below:



## 2.5 – Setting DIC Parameters

The DIC parameters can be set by going to *Analysis > Set DIC Parameters*. The main DIC algorithm used in Ncorr is based off of Bing Pan's RG-DIC framework. RG-DIC is highly robust and computationally efficient as well. More details can be found in Bing Pan's paper on RG-DIC as well as the "DIC Algorithms" section on the ncorr.com website.

A GUI should appear as shown below:



**NOTE:** A subset spacing of 3 is shown above to better illustrate the spaced points on the zoomed in subset, but for the example analyzed in this section, a subset spacing of 1 was used to obtain higher resolution results.

There are several key components to this GUI. The first is obviously the menus on the left, but it is also important to note that the subset preview is interactive. A green *impoint* (highlighted by a red square) is placed in the axes labeled "Subset Location." This point is draggable and is the center point of the subset shown on the right. The subset on the right gives an idea of what the subset spacing (space between the two dots within the red squares) and subsets will appear like. It's important to note that these highlighted points are where the subset locations will be, and not part of the speckle pattern in the uploaded image.

**Subset Options:** These options are the main components of the DIC analysis. They dictate how large the subsets should be and the spacing between them. The spacing component is purely for reducing computational load. There are defaults for both options, but it's up to the user to select the most optimal settings. The most important option to get correct is the subset radius. There is an abundance of literature available for the selection of the subset size, as well as the effects of subset size on DIC

analysis, but most of the conclusions from these studies are often based on heuristics and empirical observations. Overall, the main idea is to select **the smallest subset possible which does not result in noisy displacement data** (as larger subsets tend to have a smoothing effect). Some iteration may be required to get this option right.

**Iterative Solver Options:** The iterative solver used in the DIC analysis is the inverse compositional image alignment technique (A good paper on this topic is Baker et al and Bing Pan's paper on his adaptation for this technique for DIC). The exit criteria for this iterative solver are the norm of the difference vector as well as the number of iterations. The default options for the iterative solver are actually pretty strict, but they can be relaxed if faster analysis is desired by *reducing* the iteration number cutoff as well as *increasing* the norm of the difference vector cutoff.

**Multithreading Options:** Ncorr has the ability to use multithreading to speed up the computation process. The default number of threads is the number of cores specified by the user during installation, although I have left the option to alter this number again through this GUI.

**High Strain Analysis:** In the example used specifically in this section, high strain analysis is not required. However, the high strain analysis works by updating the reference image (as well as the ROI) and then “adding” displacement fields together. If you enable this parameter, there are two different ways the updating can proceed: In the case of the “seed propagation” option, the reference image will be updated based on the correlation coefficient and the number of iterations to convergence of the seeds. If these exceed certain heuristic thresholds, then the reference image will be updated. For the “leap frog” approach, the user can manually select how many images to use before updating the reference image. For both options, if the “automatic propagation” checkbox is enabled, then the seeds will be placed automatically for the updated reference image based on their previous displacements and the DIC analysis will proceed automatically until completion. If this is unchecked, then the user will have to replace the seeds manually every time the reference image is updated.

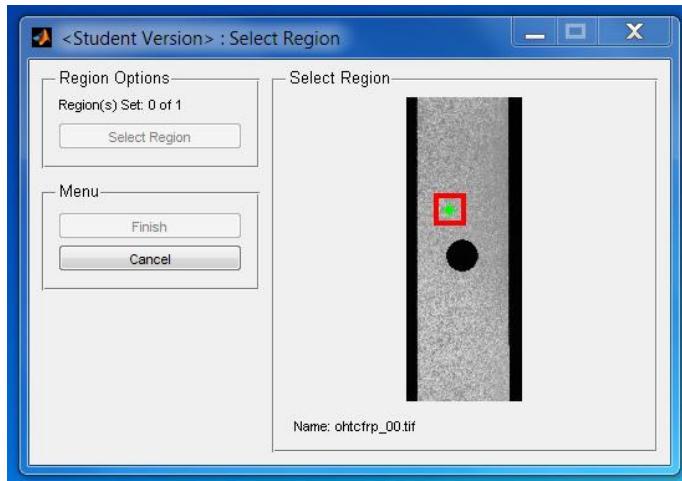
**Discontinuous Analysis:** For this example, discontinuous displacements are not anticipated, so this feature isn’t used. Regardless, subset truncation is a feature that prevents subsets from wrapping around a crack tip which can cause distortions near the cracktip. This feature is elaborated on in the discontinuous displacements section.

## 2.6 – DIC Analysis

The actual DIC analysis is performed by going to *Analysis > Perform DIC Analysis* in the main Ncorr GUI. The first step is to select a region as shown below.

### Select Region

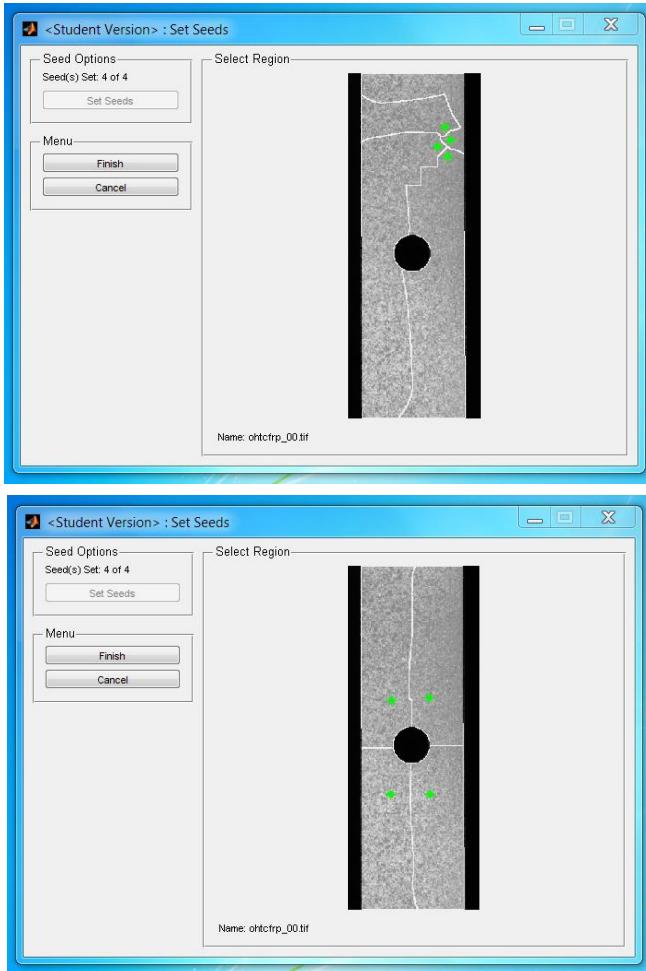
This step involves selecting a contiguous region to process. In this example, only one region will be processed (because only one is present), but it is possible to have more than one region when forming the ROI. Press the "Select Region" button and place the *impoint* anywhere within the region to select it for processing, as shown below:



### Seed Placement

The seed placement process serves three purposes: 1) it provides initial guesses for the RG-DIC analysis, 2) it partitions the ROI so that each partition can be calculated in parallel, and 3) in the case of the high strain step analysis (with seed propagation selected), it updates the reference image based on certain heuristic thresholds for the iterations to convergence and correlation coefficients of the seeds. Thus, the placement of the seeds is very important. To satisfy the first requirement, seeds must be placed such that they do not go outside the field of view (FOV) during deformation. For the second requirement, seeds should be placed such that the ROI is partitioned evenly. For the third requirement, seeds should be placed in regions of high deformation such that the reference image updates appropriately. The first two points are addressed here, while the third is addressed in the high strain section.

An example of inappropriate vs appropriate seed placements for the example reference image is shown below:

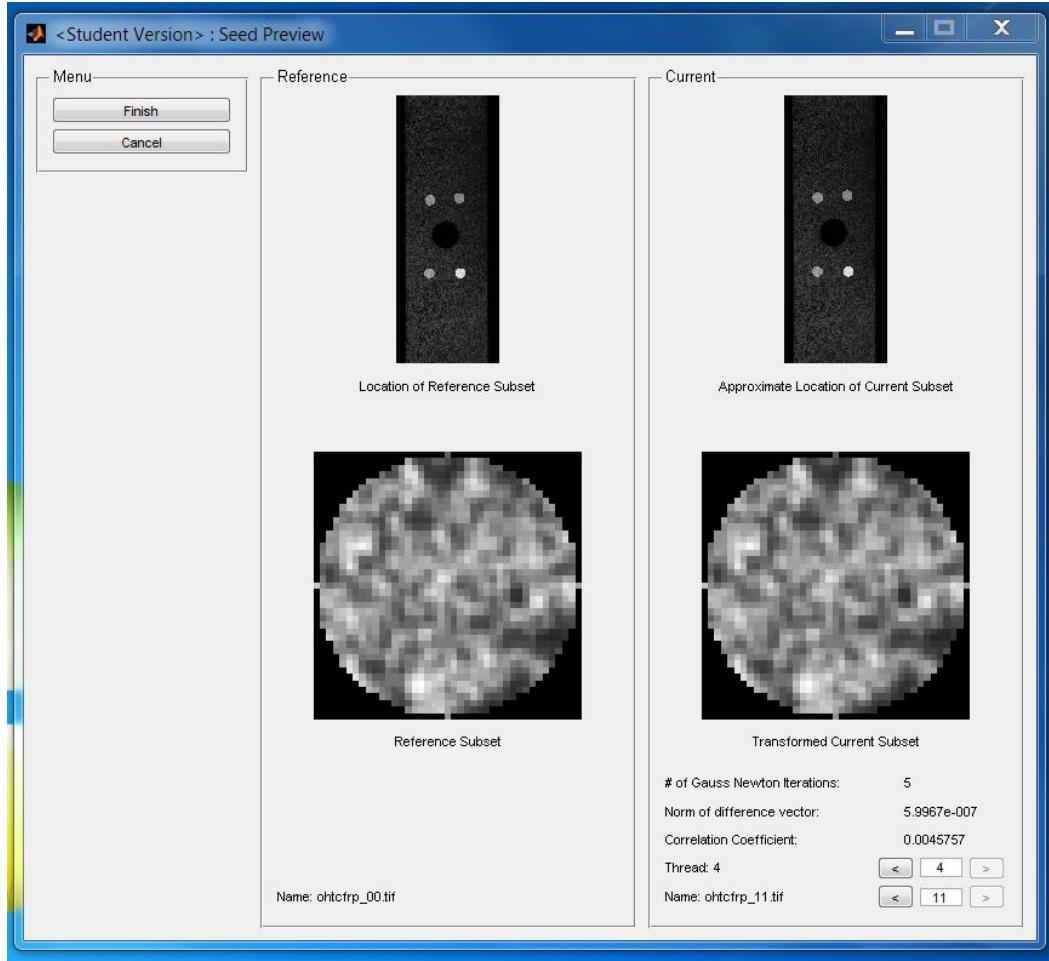


These seed positions are not preferable because the regions aren't evenly sized and they aren't nice "full" shapes.

These seed placements are more preferable because the regions are evenly sized and have nice "full" shapes.

## Seed Preview

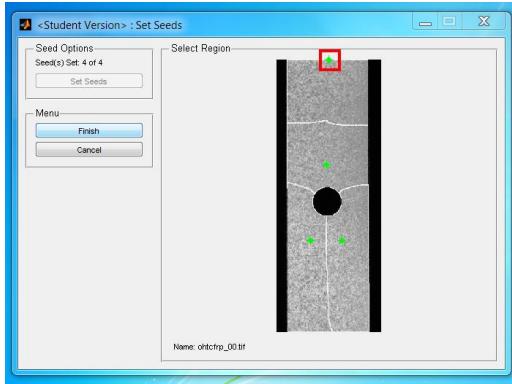
After the seed placement is done, a preview will appear. It's important that the user checks the seeds through this GUI because it's possible that the seeds can be processed incorrectly. The main reasons for seeds being incorrect can be either through a failure in convergence (this rarely happens as long as there are no large rotations or deformations present), or if the seeds travel outside the current image as the sample deforms. An example for proper seed preview is shown below:



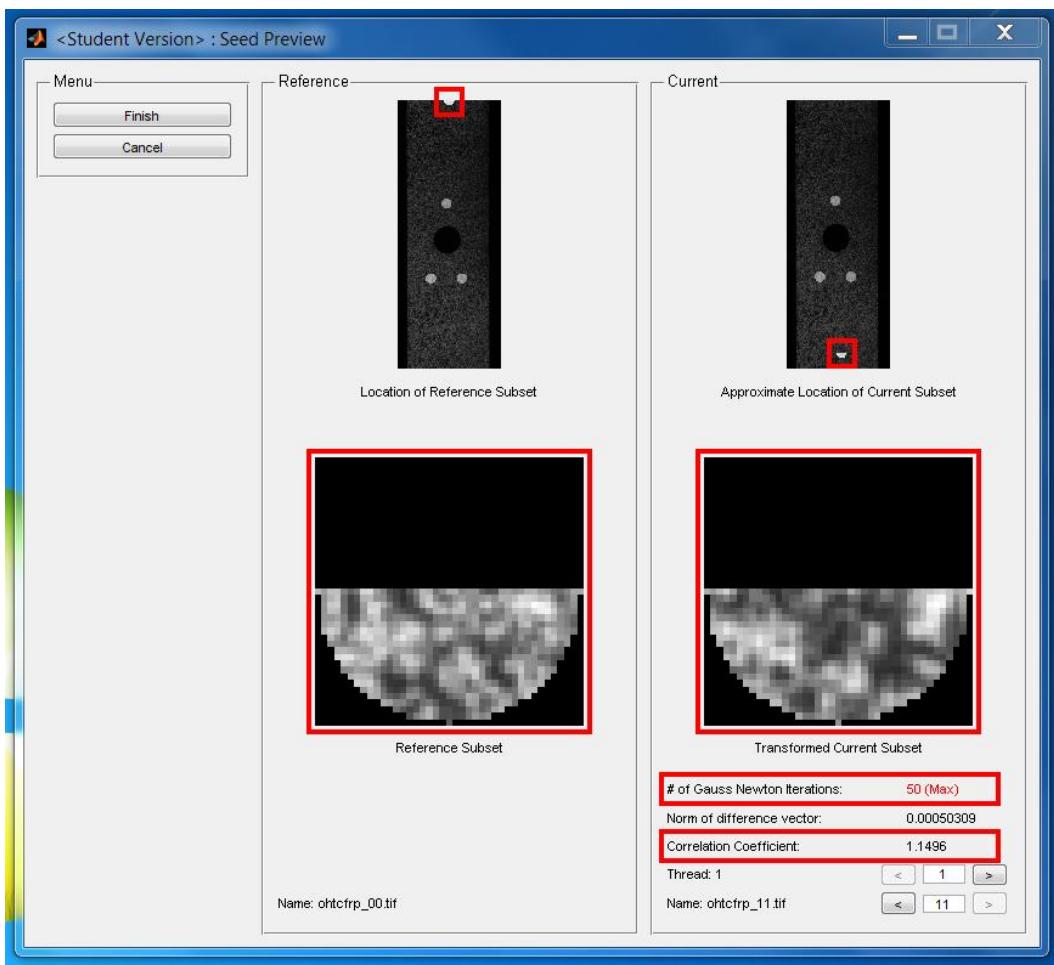
Notice that all the seed locations in the reference image seem to match appropriately with the locations in the current image. Furthermore, the reference subset and transformed current subset look very similar. Lastly, the number of iterations to convergence was way below the cutoff, and the relatively low correlation coefficient both imply a good and correct seed placement.

## Debugging

As mentioned before, it's possible for seeds to travel outside the image during deformation which results in improper seed calculations. Because this is actually a somewhat common occurrence (especially when the whole sample cannot be imaged within the field of view), an example is given below with tips on how to catch this mistake and correct it:



Suppose for instance that the seeds are placed as shown above. It turns out that the sample in the picture undergoes some translation upwards as it deforms. The seed highlighted in red actually goes "out of the picture" in some of the current images. In this case, an error message will generally display that notifies the user of a seed point with a high correlation coefficient value. This is an important indicator of a seed that has possibly been processed incorrectly. You can confirm this by viewing the seed preview below:

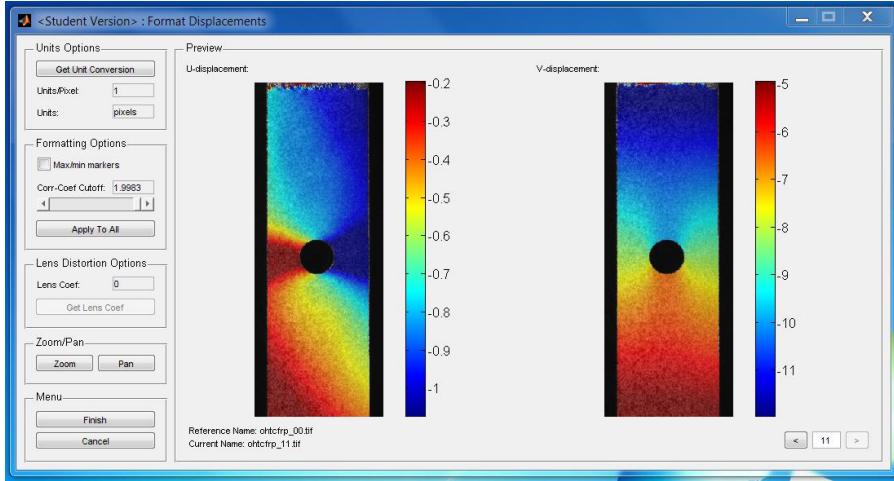


All of the highlighted things in the figure above indicate potential problems. In the location pictures, it becomes evident that the location of the top seed in the current image is completely off, whereas the other three seeds are correct. Visual inspection of the enlarged reference and current subsets also show that they look very different. Lastly, the maximum number of iterations (50 for this case) was used before convergence was reached; this is typically another good indicator of an improperly calculated seed. The solution here is to move the seed to a lower location or potentially redraw the ROI to exclude the top region.

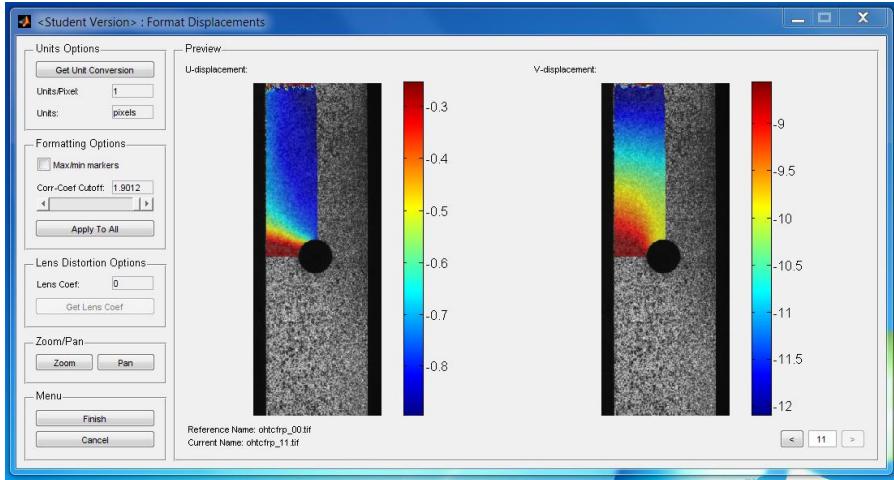
Anyway, after the seeds have been properly placed and calculated, the RG-DIC analysis will run until completion. The next step in the process is to format the displacements.

## 2.7 - Formatting Displacements

The displacement formatting begins by going to *Analysis > Format Displacements* in the main Ncorr GUI. The resulting GUI for formatting the displacements should look like this:



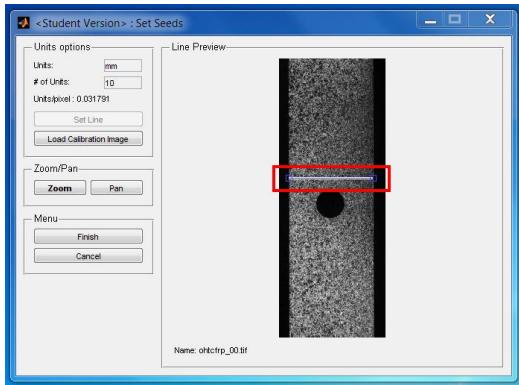
This is a good check to make sure OpenMP is working correctly. If OpenMP installed *incorrectly* and did not result in a compiler error, the analysis may still run but will result in something that looks like this:



These results are obtained because only one thread (and thus one seed) was processed. If this happens, go to *File > Reinstall* to recompile the MEX files. If problems persist, try to either install Ncorr manually, or simply use single threaded performance.

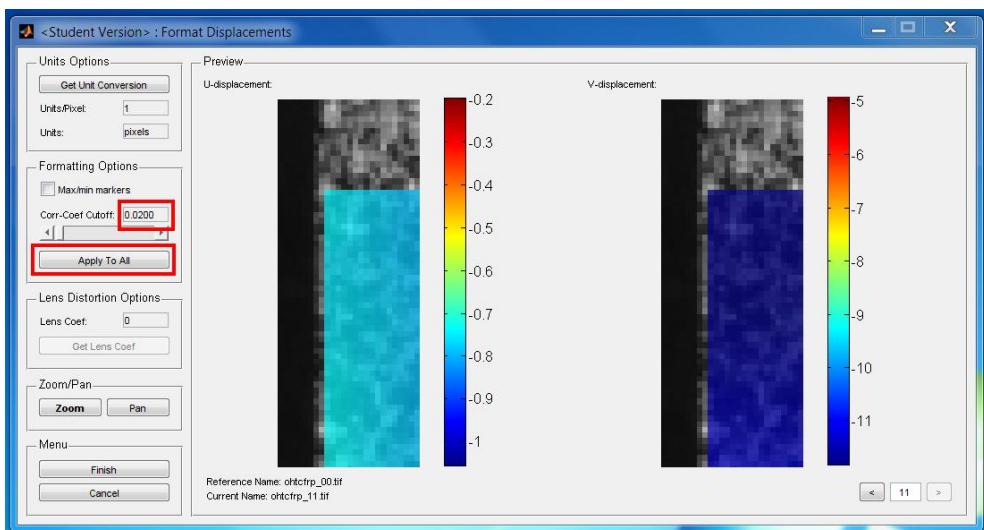
**Units Options:** These options are used to convert the displacements from pixels to real units. You can either input the value directly if you know it, or you can click the "Get Unit Conversion" button in order to load an image with a known dimension. Most tests are done with a regular camera on a sample, so the easiest way to take advantage of this feature is just to, after the experiment is complete, replace the

sample with a ruler so you can measure what the unit conversion is. An example of the GUI is shown below:



For this sample set, I'm actually not sure what the unit to pixel conversion is, so I've left the units as pixels. But, just for the sake of an example, assume that the undeformed sample is 10 mm wide. You can apply this knowledge by loading the reference image through "Load Calibration Image" and then setting a line across the sample with "Set Line." Lastly, the units (mm) and the number of units (10) can be set in their appropriate boxes. Assuming the sample is 10mm wide, this will apply the correct conversion to convert the displacements from pixels to mm (determined to be 0.031791 mm/pixel in this case). It's important to note that this assumes pixels are square, which is usually the case.

**Formatting Options:** These options help filter out "bad" data points. In the example done in this section, some of the points in the ROI travel outside the FOV. These points can be filtered out by reducing the correlation coefficient cutoff. In conjunction with this, the locations of the min and max displacement values can assist in this analysis, as "bad" data points generally have very high or very low values. A good cutoff for the example done in this guide is to set the correlation coefficient cutoff to 0.02 (which was applied to all images), as shown below:



Note that the preview above was zoomed in using the zoom/pan options. The filtering out of bad data points is shown below:



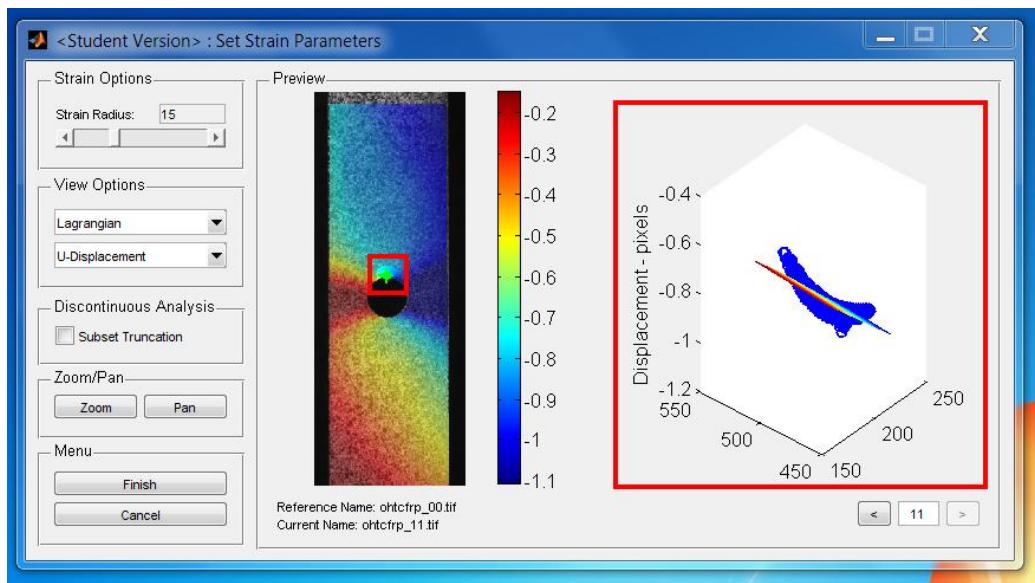
**Lens Distortion Options:** This is an option to correct for errors involving radial lens distortion. The distortion correction follows from eq.6 in “Systematic errors in two-dimensional digital image correlation due to lens distortion.” The correction is applied assuming the distortion center is at the center of the image; this might not always be true but is usually a reasonable assumption. If no calibration tests have been done or you’re unsure what the lens coefficient is, then just leave it as 0.

As of now, a lens coefficient can only be applied. Bing Pan provides a method for determining the lens coefficient, but I have yet to integrate this into Ncorr. The process basically requires DIC analysis to be run on images taken during rigid body translation tests and then performing a simple least squares analysis to solve for the lens distortion parameter. In future revisions of the program, I may implement this feature.

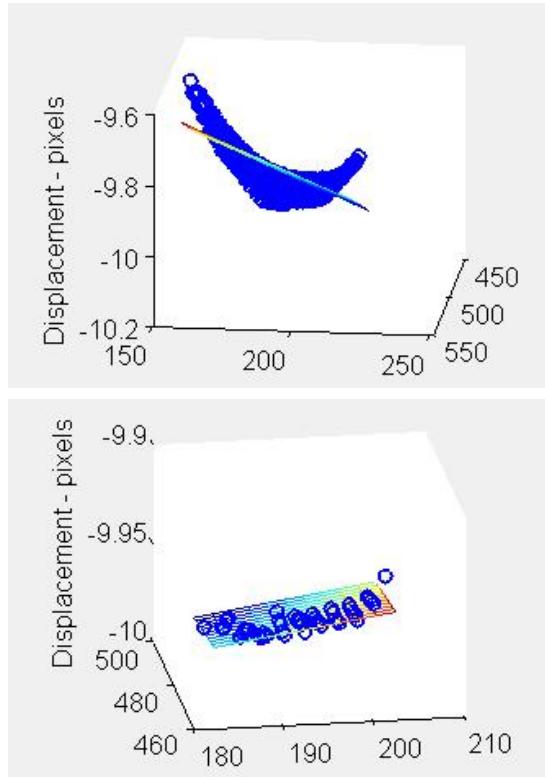
## 2.8 - Strain Analysis

The strain analysis begins by going to *Analysis > Calculate Strains* in the main Ncorr GUI. The strains are calculated from the displacement data by using a least squares plane fit to a local group of data points (in this case a contiguous circle) which is based on Bing Pan's work on strain calculation. The displacement gradients are then found from the plane parameters; these gradients are used to calculate **Green-Lagrangian** and **Eulerian-Almansi** strains. A more detailed description is given in the "DIC Overview" section on the ncorr.com website.

**Strain Options:** The only parameter you can vary here is the strain radius. This is the radius of a circle which selects a group of points to fit a plane to. A preview is provided so the user can visualize the plane fitting. The selection of the ideal strain radius is similar to the selection of the ideal subset radius, in that **the smallest radius is desired which does not result in noisy strain data**. The default radius is set to 15, but it is up to the user to select the most optimal radius for their data.



The *impoint* highlighted in the red box is dragable, and updates the plane fit shown on the right side. The recommended way to use this analysis is to drag the point to areas of high deformation and view if the curve fit is still reasonable. An example of this is shown below:



Visually, it appears the strain radius is too large for this displacement field.

Reducing the strain radius to a value of 5 appears to give a much better fit for the plane. In this example, this value is used to generate the strain field.

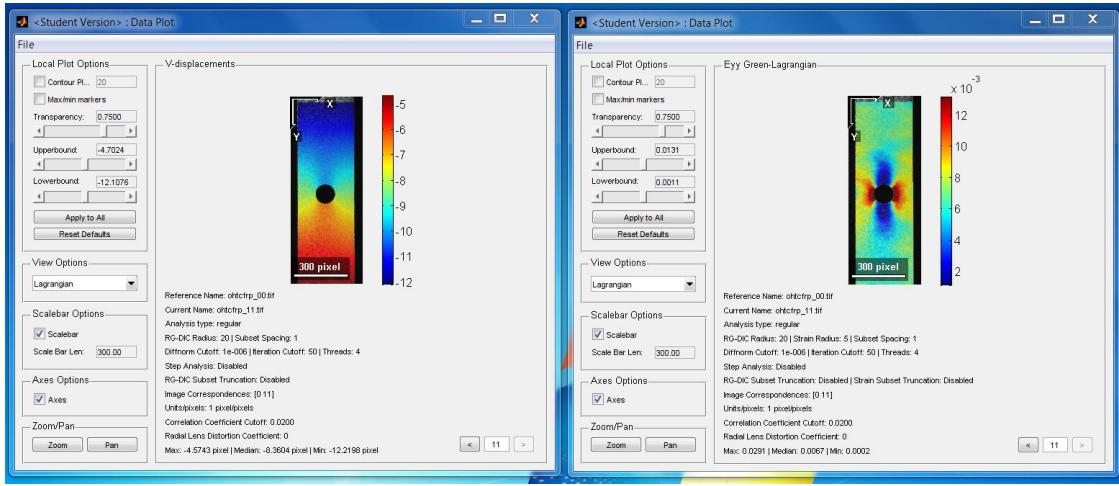
Lastly, the strain radius should be selected with regards to the last current image (which is shown first by default), as this image will most likely have the highest displacement gradients.

**View Options:** These dropdown menus allow you to switch from the U-displacements to the V-displacements and vice versa. They also allow you to switch from the Lagrangian to the Eulerian perspective. By default, both Green-Lagrangian and Eulerian-Almansi strains are calculated.

**Discontinuous Analysis:** Just like for DIC, there is also an option to utilize subset truncation for the strain calculations. This feature prevents subsets from wrapping around a crack tip which can cause distortions near the cracktip. This feature is elaborated on in the discontinuous displacements section.

## 2.9 - Plotting

At this point, it is assumed either the displacements have been formatted or the strains have been calculated. Plotting can be achieved by going to *Plot > View Displacement Plots* or *Plot > View Strain Plots* in the main Ncorr GUI. This is the last step of the analysis. An example of the plots used in the examples in this user guide is shown below:



**NOTE:** From my experiences working with the plotting tool on different operating systems and versions of MATLAB, there appear to be inconsistencies between versions with regards to the stacking order of objects (i.e. sometimes the scalebar/axes appear behind the data if a contour plot is present), but I tried to optimize it for Windows on version R2012a. If the axes, scalebar, and contour plot are disabled, the plot appears to work fine on all versions of MATLAB I've tested Ncorr on.

There are a couple basic features available, the first one being that the plot windows are resizable. They are the only GUI windows in Ncorr that are resizable at the time of writing this manual. Furthermore, if you hover your cursor over the data, a data cursor appears that informs you of the displacement or strain data at that location, as shown below:



The “x-pos” and “y-pos” positions are the integer locations of the cursor *with respect to the actual data array*. This was done for convenience so the user can access the value in the array very easily. For example, in the figure above, the last V displacement plot is shown. You can access this point directly using:

```
handles_ncorr.data_dic.displacements(11).plot_v_ref_formatted(317,163)
```

The resulting output of the terminal will be -7.5719, which matches the V displacement shown. The index for the plot is the same that is shown between the “<” and “>” buttons on the bottom right of the plotting GUI. For more information on the “data\_dic” structure, please consult section 2.10.

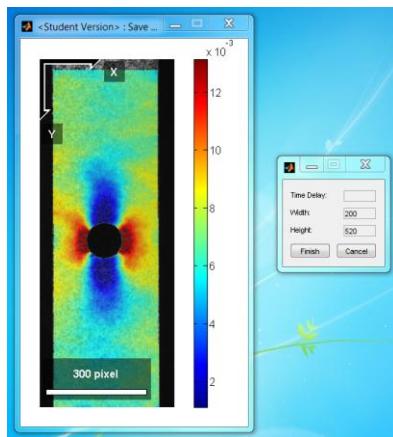
**Local Plot Options:** These options are pretty self-explanatory. The only thing to note here is that these options only affect the plot that is being adjusted (as opposed to other options which affect all open plots). Furthermore, the transparency option is disabled when the contour plot is enabled. This is because transparency does not work well with the contour plot. In general, the contour plot does not work that well with strain plots because strain plots are generally noisier than displacement plots which results in jagged contours.

**View Options:** Once again, these options are pretty self-explanatory. These allow you to switch between Lagrangian and Eulerian perspectives. For strains, these will switch between Green-Lagrangian strains and Eulerian-Almansi finite strains tensors. This option will affect all open data plots.

**Scalebar Options:** This affects the visibility and length of the scalebar (displayed in the bottom-left of the plot). This option will affect all open data plots.

**Axes Options:** This option affects the visibility of the axes (displayed in the top-left of the plot). This option will affect all open data plots.

**Saving the Figure:** A figure can be saved by going to *File > Save Image > Save Image Without Info*, *File > Save Image > Save Image With Info*, or *File > Save Gif within the plot window*. This will allow you to save nice figures for publications, websites, or any other purpose. In this guide, I will focus on saving images, as the animated gif option is more useful for high strain calculation (which is discussed in the high strain section). After selecting the save image option, a menu with a preview should appear as so:



This will allow you to resize the image if you desire. Please note that if a large spacing parameter is used, the image may look blurry if you upsize it because the image and data plot must be the same size when plotting in MATLAB. The best solution for this would be to rerun the analysis with a smaller subset spacing parameter. Also, I recommend saving the figure as a “.tif” because MATLAB appears to compress other image formats which may lead to a loss of quality.

## 2.10 - Directly Obtaining and Altering DIC Data

The DIC data can be accessed directly through "handles\_ncorr." For example, by typing "handles\_ncorr.data\_dic" into the MATLAB terminal, you can access the data structure that contains the strain and displacement data as shown below:

```
EDU>> handles_ncorr.data_dic
ans =
    displacements: [1x11 struct]
    dispinfo: [1x1 struct]
    strains: [1x11 struct]
    straininfo: [1x1 struct]
```

Notice that both "dispinfo" and "straininfo" only have one element. This is because they contain parameters which are consistent over the entire analysis. On the other hand, both "displacements" and "strains" have eleven elements. This is because these fields contain information specific to each current image. For the example used in this section, we have uploaded eleven current images, which correspond to eleven elements seen above.

The size of the displacement/strain plots themselves depend on the image and the subset spacing utilized. For the Lagrangian perspective, the size of the displacement/strain plots will be:

```
size(img_ref(1:1+subsetspace:end,1:1+subsetspace:end))
```

where "img\_ref" is the 2D reference image grayscale array and "subsetspace" is the subset spacing parameter (where a value of 0 results in no reduction). For example, if a reference image of size 400x1040 and a subset spacing of 1 is used (as is done in the example in this manual), then the Lagrangian displacement/strain plots will be 200x520.

The displacements with respect to the reference configuration (Lagrangian) are stored in the plots with the "\_ref" suffix. The "\_cur" suffix is used for displacements with respect to the current configuration (Eulerian).

Each field of "handles\_ncorr.data\_dic" will be explained in a table format as shown below:

<b>handles_ncorr.data_dic.displacements</b>	
Field	Explanation
plot_u_dic	Plot containing U displacement data WRT the updated reference image (based on imgcorr) after DIC analysis. This plot is used for computational purposes.
plot_v_dic	Plot containing V displacement data WRT the updated reference image (based on imgcorr) after DIC analysis. This plot is used for computational purposes.
plot_corrcoef_dic	Plot containing correlation coefficient data WRT the updated reference image (based on imgcorr) after DIC analysis.
roi_dic	ROI corresponding to valid displacement data WRT the updated reference image (based on imgcorr) after DIC analysis.

plot_u_ref_formatted	Plot containing U displacement data WRT the reference configuration after it has been formatted. This plot is displayed in the plot figure. This plot is displayed in the plot figure.
plot_v_ref_formatted	Plot containing V displacement data WRT the reference configuration after it has been formatted. This plot is displayed in the plot figure.
roi_ref_formatted	ROI corresponding to formatted displacement data WRT the reference configuration
plot_u_cur_formatted	Plot containing U displacement data WRT the current configuration after it has been formatted. This plot is displayed in the plot figure.
plot_v_cur_formatted	Plot containing V displacement data WRT the reference configuration after it has been formatted. This plot is displayed in the plot figure.
roi_cur_formatted	ROI corresponding to formatted displacement data WRT the current configuration

<b>handles_ncorr.data_dic.dispinfo</b>	
Field	Explanation
type	String representing whether "regular" or "backward" analysis was done; this value can either be 'regular' or 'backward'
radius	Integer representing the subset radius
spacing	Integer representing the subset spacing
cutoff_diffnorm	Number representing the cutoff for the norm of the difference vector in the inverse compositional method
cutoff_iteration	Integer representing the cutoff for the maximum number of iterations allowed for the inverse compositional method
total_threads	Integer representing the total number of threads specified during the RG-DIC parameter specification stage (can be different than the number of cores specified during installation)
stepanalysis	Structure containing data for the step analysis. The fields "enabled," "type," "auto," and "step" represent if the step analysis is enabled, whether seed propagation or leapfrog is used, whether seeds will automatically get placed at each step, and the step parameter for the leapfrog analysis, respectively.
subsettrunc	Logical parameter specifying if subset truncation was used for the RG-DIC analysis.
imgcorr	Structure containing the image correspondences for the step analysis. For each element, there is an "idx_ref" and "idx_cur." The idx's are zero based and assume the reference and current images are concatenated. idx_ref is the index for the updated reference image whereas idx_cur is the index of the last current image WRT this reference image.
pixtounits	Number representing the conversion between pixels to real units. In the form of units/pixels.
units	String containing the units
cutoff_corrcoef	Number representing the minimum correlation coefficient value allowable for displacement datapoints corresponding to a dataplot.
lenscoef	Number representing the radial lens distortion coefficient

<b>handles_ncorr.data_dic.strains</b>	
Field	Explanation
plot_exx_ref_formatted	Plot containing the Exx Green-Lagrangian strains (WRT the reference image). This plot is displayed in the plot figure.
plot_exy_ref_formatted	Plot containing the Exy Green-Lagrangian strains (WRT the reference image). This plot is displayed in the plot figure.
plot_eyy_ref_formatted	Plot containing the Eyy Green-Lagrangian strains (WRT the reference image). This plot is displayed in the plot figure.
roi_ref_formatted	ROI containing information for valid points after strain calculation (It's possible for some points to be removed during strain calculation for regions that do not have enough points to form a plane fit)

plot_exx_cur_formatted	Plot containing the Exx Eulerian-Almansi strains (WRT the current image). This plot is displayed in the plot figure.
plot_exy_cur_formatted	Plot containing the Exy Eulerian-Almansi strains (WRT the current image). This plot is displayed in the plot figure.
plot_eyy_cur_formatted	Plot containing the Eyy Eulerian-Almansi strains (WRT the current image). This plot is displayed in the plot figure.
roi_cur_formatted	ROI containing information for valid points after strain calculation (It's possible for some points to be removed during strain calculation for regions that do not have enough points to form a plane fit)

<b>handles_ncorr.data_dic.straininfo</b>	
Field	Explanation
radius	Integer representing the strain radius
subsettrunc	Logical parameter specifying if subset truncation was used for the strain analysis.

Lastly, any information that the user wants to alter should be copied first to another variable in the main workspace. By default, the properties of Ncorr are set to "(SetAccess = private)" to prevent inadvertent alteration of data. If the user wants to alter the data directly, then the user can go into the **ncorr.m** source code and change "(SetAccess = private)" to "(Access = public)" to alter these properties directly as shown below:

```

113 properties (SetAccess = private)
114 % GUI handles:
115 handles_gui;
116
117 % DIC data:
118 reference;
119 current;
120 data_dic;
121
122 % Installation data:
123 support_OpenMP;
124 total_cores;
125 end

```

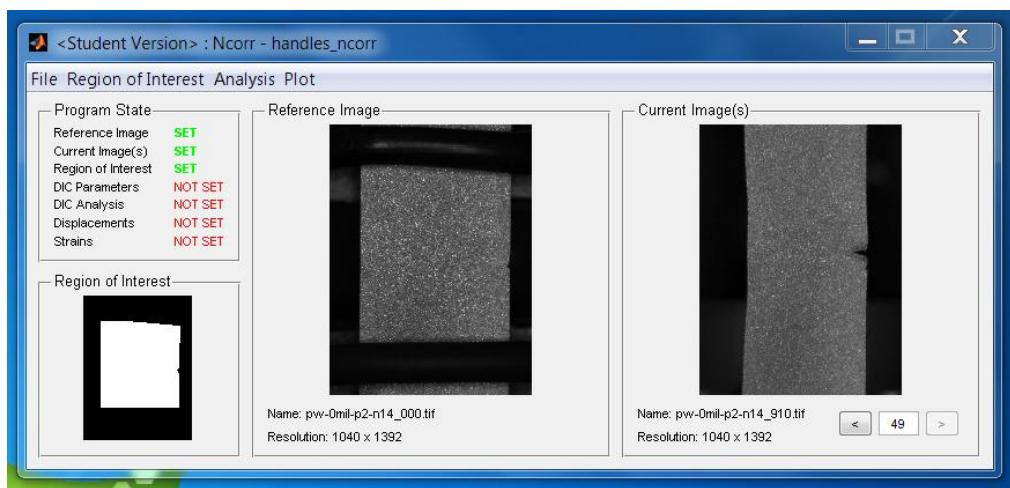
It may be possible to crash Ncorr by setting inconsistent data and thus caution is advised if going this route.

### 3.1 – High Strain DIC Analysis

For this section, it is assumed the program has already been installed and the user has already read the General DIC section. The images used for this section (the “weld” sample from SEM’s DIC challenge) are available off the Ncorr website if the user would like to follow along:

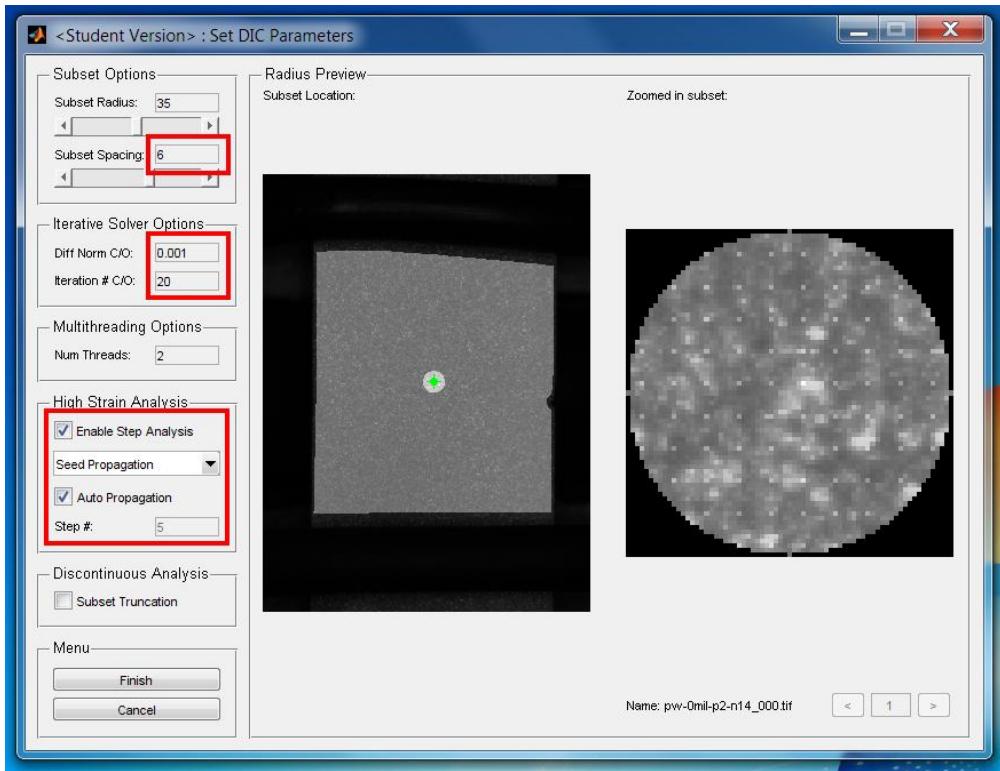
<http://ncorr.com/download/sample13.zip>

Start by setting the reference image to the first image (post fix of “000”) of the set. Next, load the current images by using “Lazy Load” (or potentially “Load All” if your computer has a lot of RAM) and selecting all the images from postfix of “020” until the postfix of “910” (note that at “915” the sample has failed- so omit this picture) as the current images. Lastly, set the *reference ROI* to the ROI provided in the image set (named “roi.tif”). If successful, the GUI should appear as below:



Now, if the user simply proceeds as is done in the general DIC section, he/she will find that the seeds for the latter images will always be incorrect for the latter images. This is because this sample undergoes very high deformation. The underlying problem here is that the pattern between the reference image and the last current image changes so much that the matching algorithm no longer works. This type of sample requires the reference image to be updated. To account for this, the high strain step analysis needs to be enabled when setting the DIC parameters; this is discussed in the next paragraph.

For high strain DIC, it’s recommended to set the type of step analysis to “seed propagation” with the “automatic propagation” checkbox enabled, as shown below:

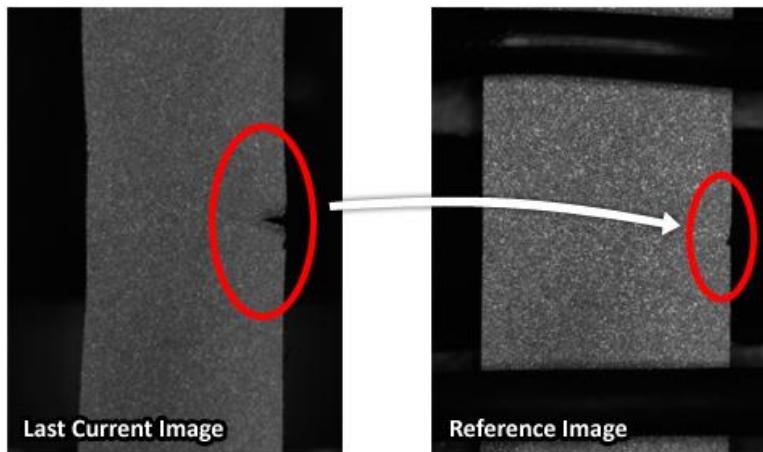


There are a couple things to note here. First of all, there are a large number of images involved in this analysis, so performance issues will be a concern. To increase the speed of the analysis, I've elected to increase the subset spacing to 6. Furthermore, the norm of the difference vector has been increased to  $10^3$  and the iteration cutoff has been reduced to 20. This will reduce the number of material points to analyze and also make the conditions on the iterative solver more relaxed. The effects of doing these three things will be a lowered resolution in the acquired displacement field, and a slightly reduced accuracy as well. If time permits, then the spacing, norm of difference vector, and iteration cutoff can be left to their default values.

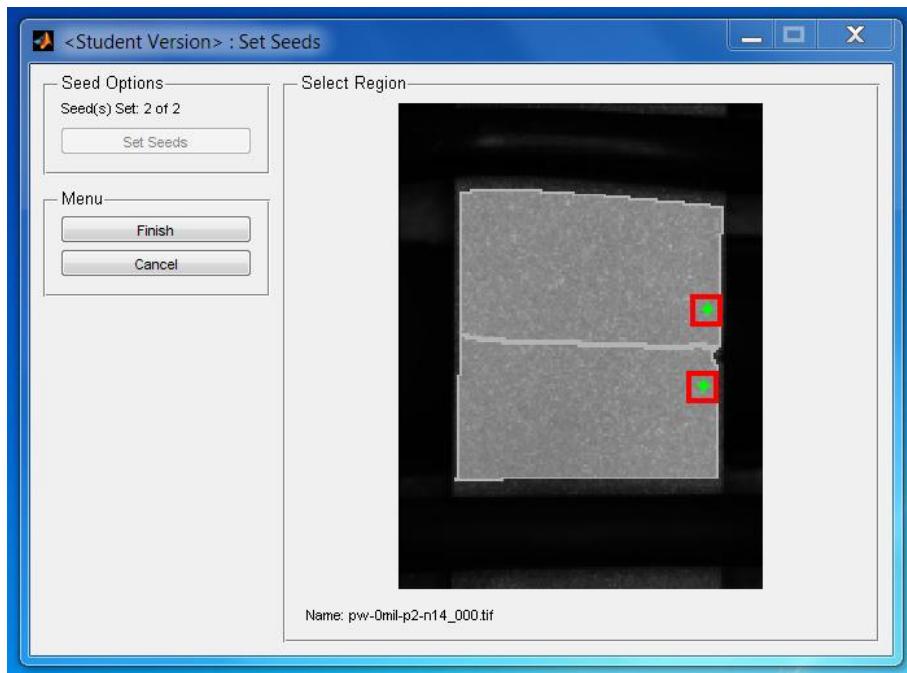
In addition, the high strain analysis has been set with the seed propagation and automatic propagation parameters enabled. This allows the program to automatically update the reference image, as well as to automatically place seeds so the analysis can proceed in an automated fashion until completion. If "leapfrog" is selected, then the reference image will update according to whatever step number is provided. Lastly, if auto propagation is disabled, then the user will have to manually place the seeds every time the reference image is updated. This can be quite tedious, but also ensures the seeds are correct, so this option should only be disabled if you want to ensure completely that the seed placements are correct.

The next crucial step is the seed placements. Recall from the general DIC analysis that there are two main considerations for seed placement for the regular DIC analysis. First of all, they need to be placed on a material point that does not exit the FOV as the sample deforms. Next, they need to be placed such that the regions are more or less evenly portioned so they can be calculated nicely in parallel. For high strain analysis with seed propagation, there's another consideration: since the reference image is

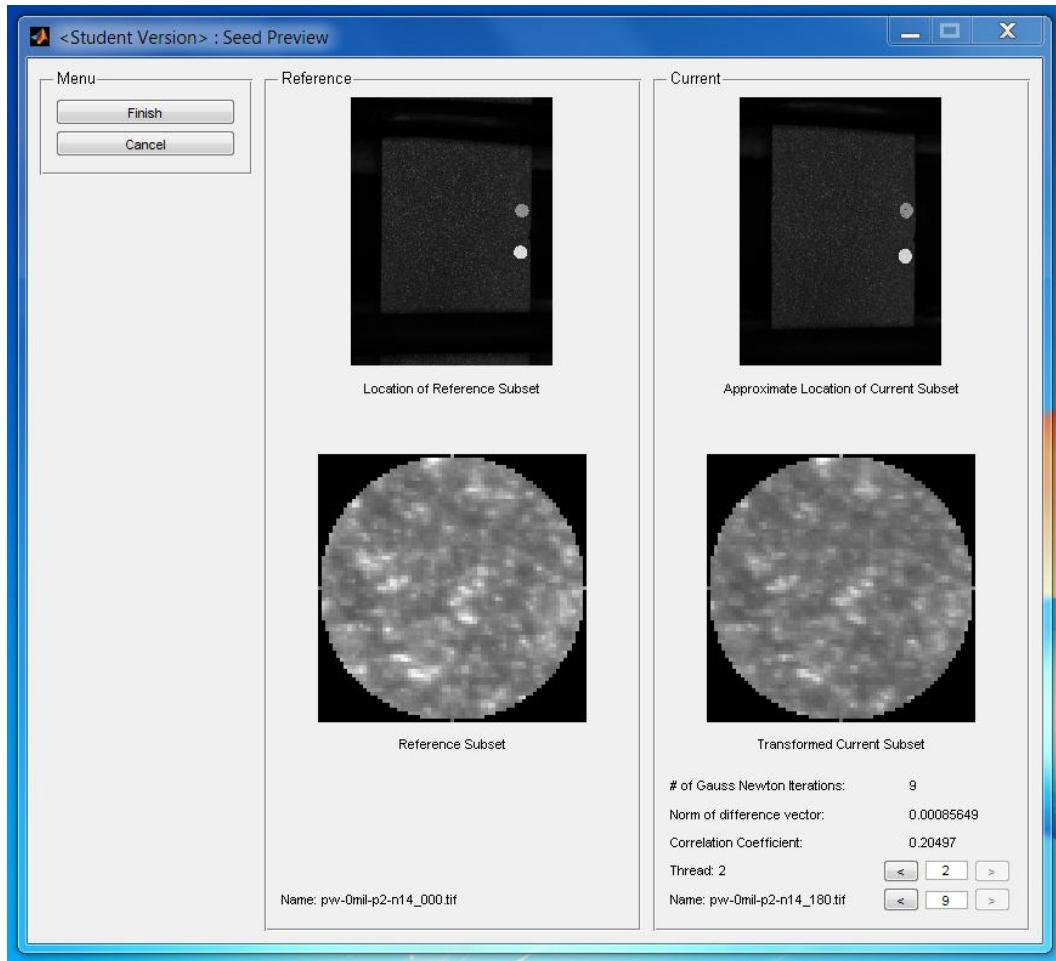
updated based on the correlation coefficient and number of iterations to convergence of the seed points, they need to be placed in a region of high deformation, so that the reference image updates correctly. If you examine the last current image in the example set, you'll see that the region of highest deformation is near the region circled below:



Basically, from inspection of the last current image (shown on the left), you can assume the highest deformation occurs approximately in the region encircled by the red ellipse (where the crack initiates). It would be ideal to place the seed points on the reference image corresponding to this region so that the reference image updates properly. With this in mind, the seeds are placed as so:

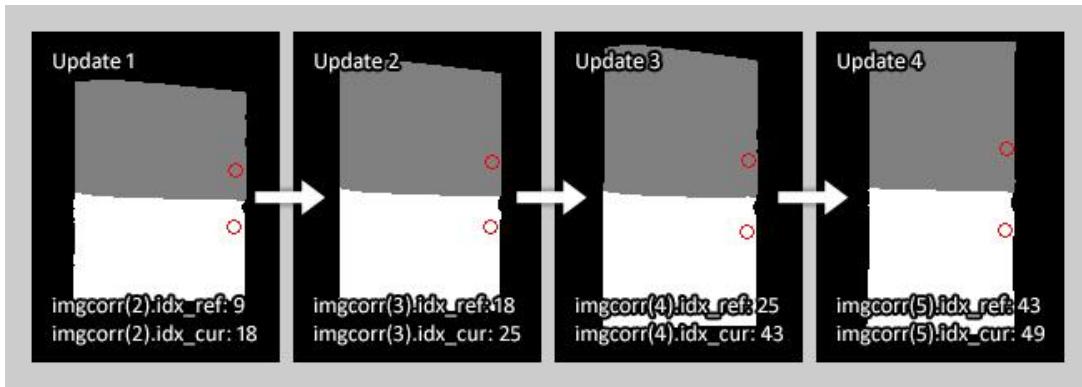


These seed locations (shown in the red boxes) satisfy the three requirements explained previously. Clicking finish yields the following seed preview screen:



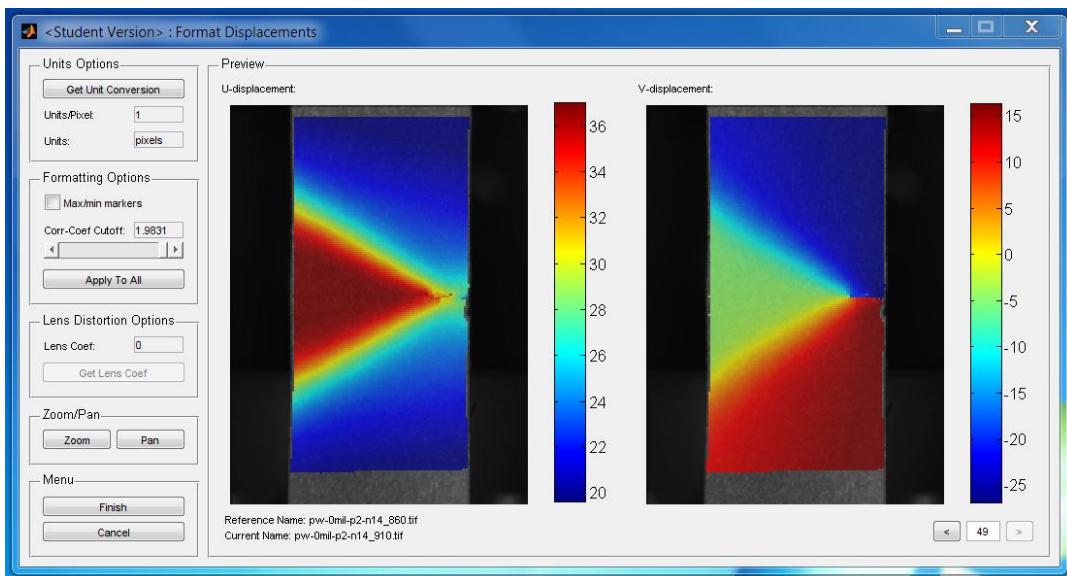
The most important thing to note here is that the last image seeded is not the last image of the series. The image with postfix “180” was determined to be the last properly seeded image of the series (Note that this can vary; it’s possible the image can be a couple before or after this one depending on the seed placement). This image will serve as the updated reference image until it is updated again. Click finish to proceed with the analysis. It should continue until completion.

Before continuing, I’d like to explain what exactly is going on “under the hood.” Basically, every time the reference image gets updated, the seeds are automatically “propagated” and placed as the analysis proceeds. Additionally, the ROI is also updated based on the displacements on the boundary of the ROI. For this example, the updated thread diagrams with seed placements are shown below:

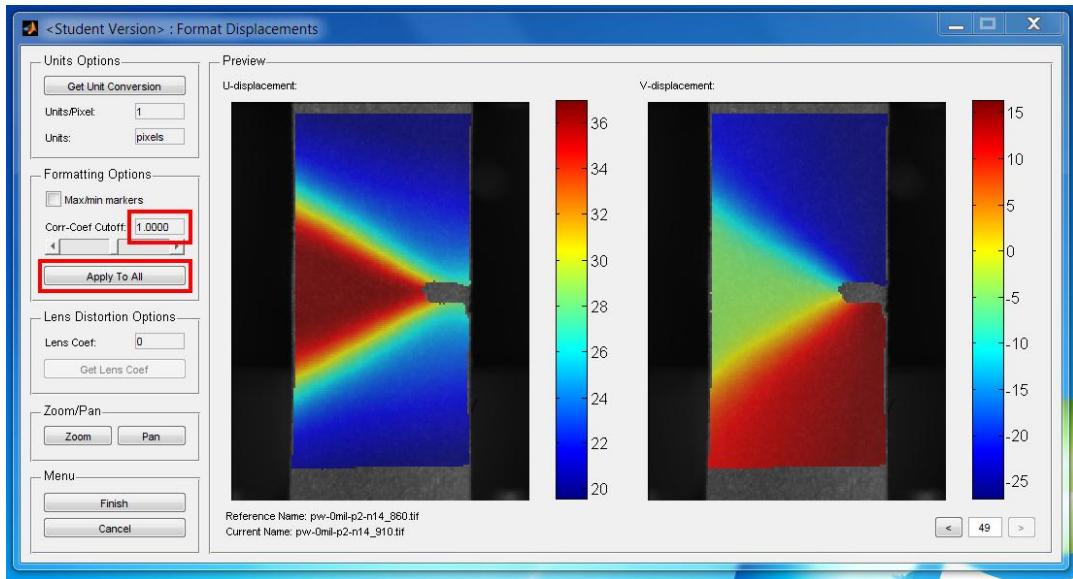


For this particular example, there were four updates. The ROIs and seed positions (shown as red circles) are automatically updated and then subsequently processed. The corresponding “imgcorr” structures are also superimposed as an example for how this structure works.

From here, the only problem is that the displacement fields are now going to be with respect to the updated reference images. The next step is to somehow “add” them in order to bring them all back to the Lagrangian perspective (i.e with respect to the original reference image). This step is taken care of when the displacements are formatted. The formatting displacements GUI should look similar to the one shown below:

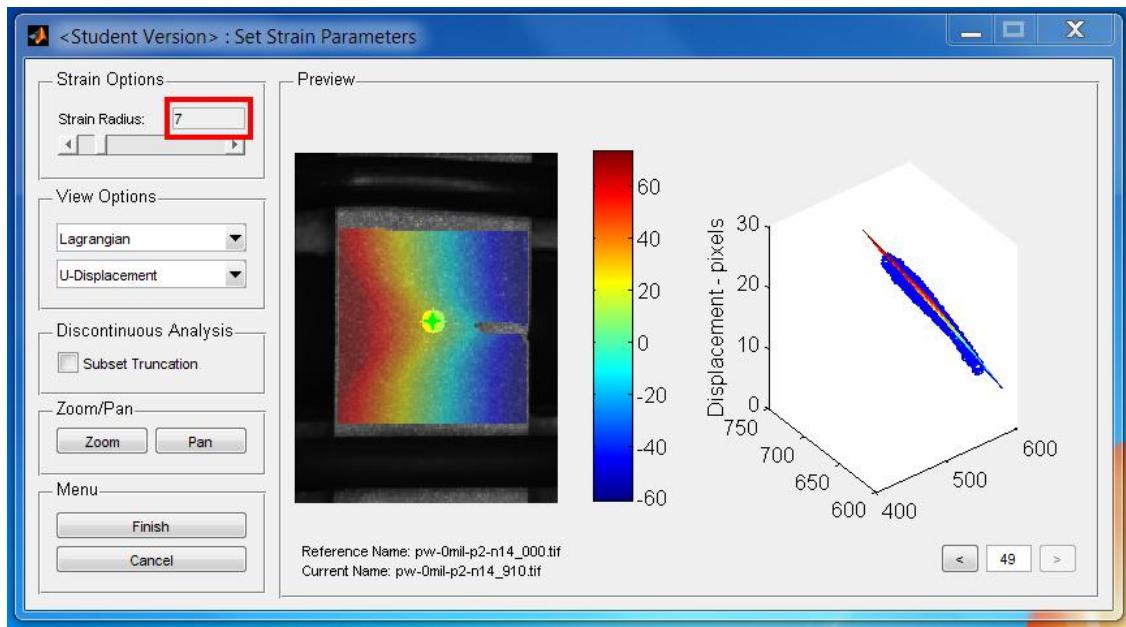


A couple things to note here: first of all, near the top of some of the displacement fields, the data appears to be very poor. This is because these material points travel outside the FOV as the sample deforms, so these points need to be removed. Furthermore, a crack develops in the sample. This means points near this crack are also poorly analyzed, so they should be removed as well. It appears that setting a correlation coefficient cutoff of 1.0 yield good results:



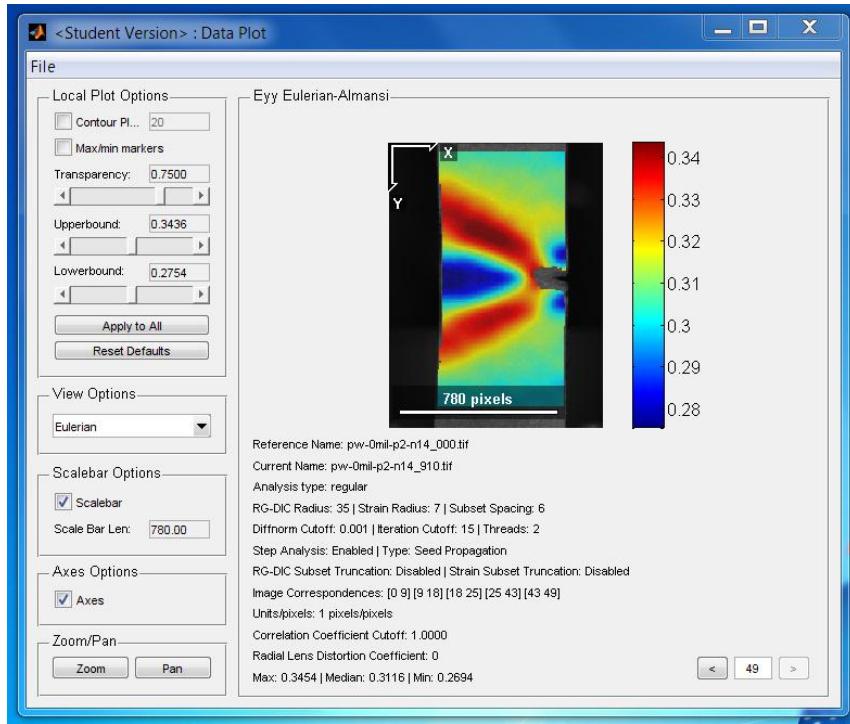
One last thing to point out is that the reference name shown above (near the bottom left) refers to the updated reference image.

The last step of the analysis is to calculate the strains. A strain radius of 7 appears to give good results as shown below:

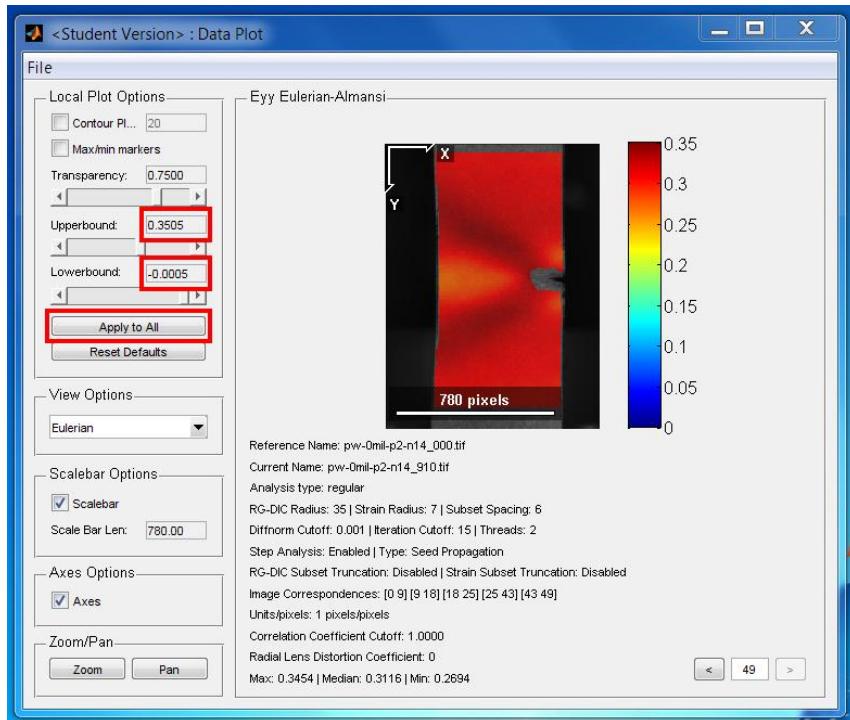


In terms of plotting, the only distinction to really make between high strain and regular DIC analysis is that the user might want to consider saving an animated gif instead of regular pictures since so many data plots are available. In this particular case, the Eulerian perspective makes for especially good animations. Furthermore, it's usually appropriate to set the same upperbound and lowerbounds for strain in all the plots.

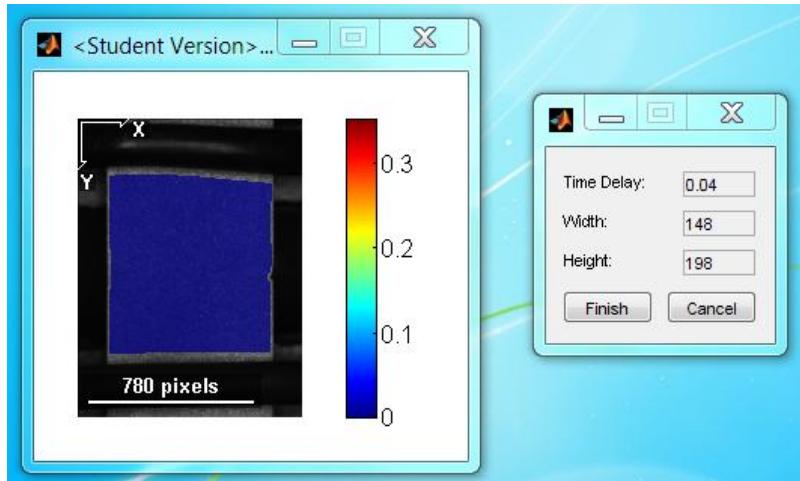
First, a plot for Eyy is shown below:



In order to set all the bounds for the strains the same, it's necessary to set them such that the bounds accommodate all plots. For this example, the lowerbound must be set near zero as shown below:



If this lowerbound is set any higher, it might result in an earlier plot becoming completely saturated. If you do this, an error message will appear. Furthermore, I tend to add or subtract 0.0005 from the bounds to guarantee the tick marks appear on the top and bottom of the colorbar as well, as shown above. Anyway, after hitting “applying to all”, select *File > Save GIF* which will bring up the following GUI:



Notice that, in addition to the size, you can also modify the time delay between animation frames. From my experience, a time delay of 0.04 (25 fps) seems to work well for this data set, but the user can always opt to increase it in order to slow the animation down and vice-versa.

## 4.1 – Discontinuous Displacements DIC Analysis

For this section, it is assumed the program has already been installed and the user has already read the General DIC section. The images used for this section (called the “cracked sample”) are available off the Ncorr website if the user would like to follow along:

<http://ncorr.com/download/cracksample.zip>

For this analysis, images from my lab are going to be used. Furthermore, to demonstrate some of the features of Ncorr, they will be loaded through the Ncorr handle instead of the GUI. The images were taken from a high resolution CCD camera and are thus quite large. To account for this, I've decided to read them through a script and then downsize them before loading them into Ncorr. The script is included with the images but is also show below:

```
%% Open Ncorr
handles_ncorr = ncorr;

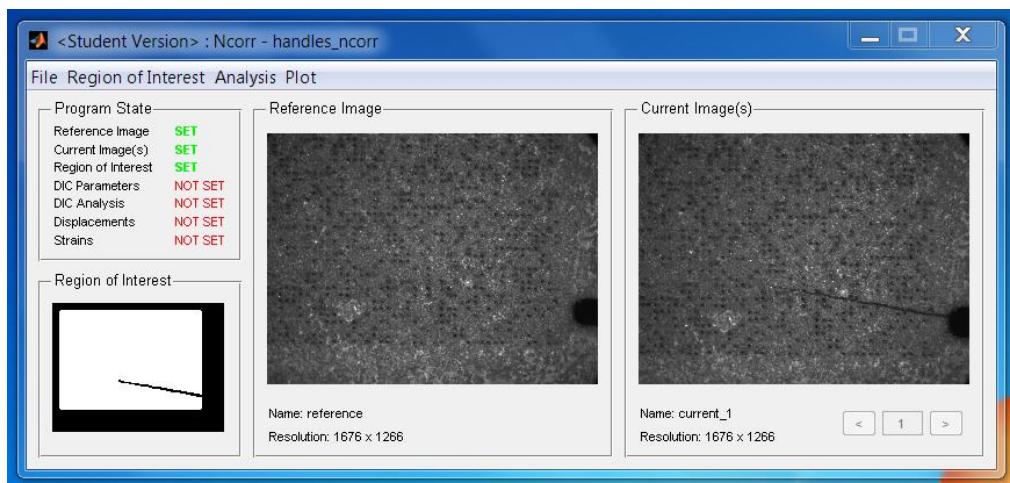
%% Set Parameters
% Load Ref
ref = imread('ref.TIF');
ref = imresize(ref,0.5,'bicubic','Antialiasing',true);

% Load Cur
cur = imread('cur.TIF');
cur = imresize(cur,0.5,'bicubic','Antialiasing',true);

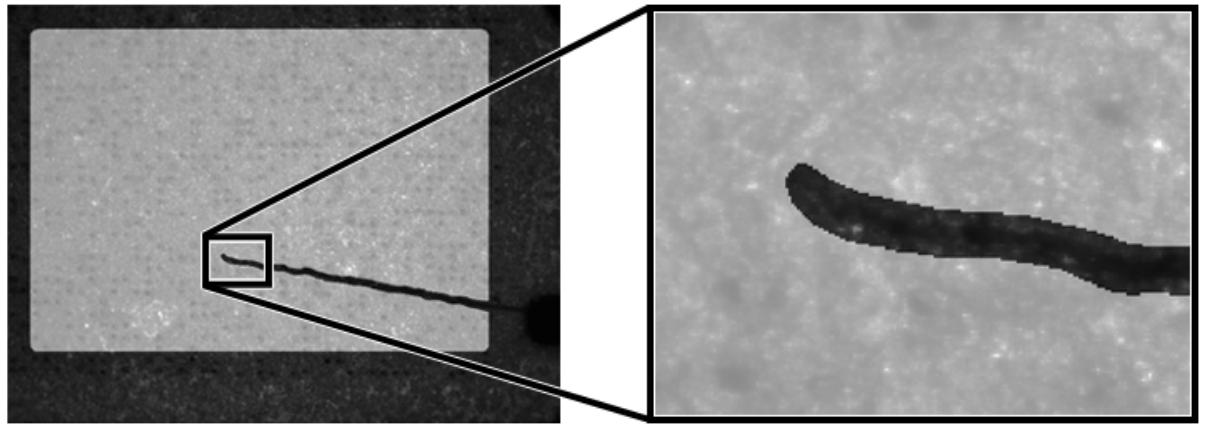
% Load ROI
roi = rgb2gray(imread('roi_half.TIF'));
roi = roi > 5;

% Set Data
handles_ncorr.set_ref(ref);
handles_ncorr.set_cur(cur);
handles_ncorr.set_roi_cur(roi);
```

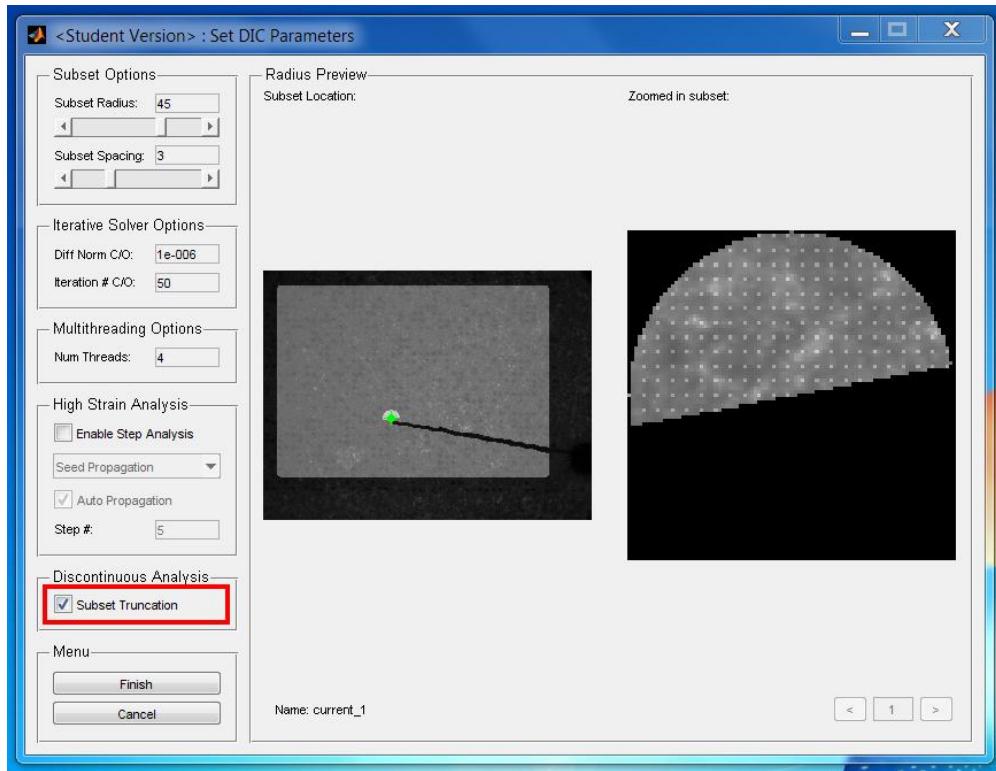
The GUI should appear as so if everything went correctly:



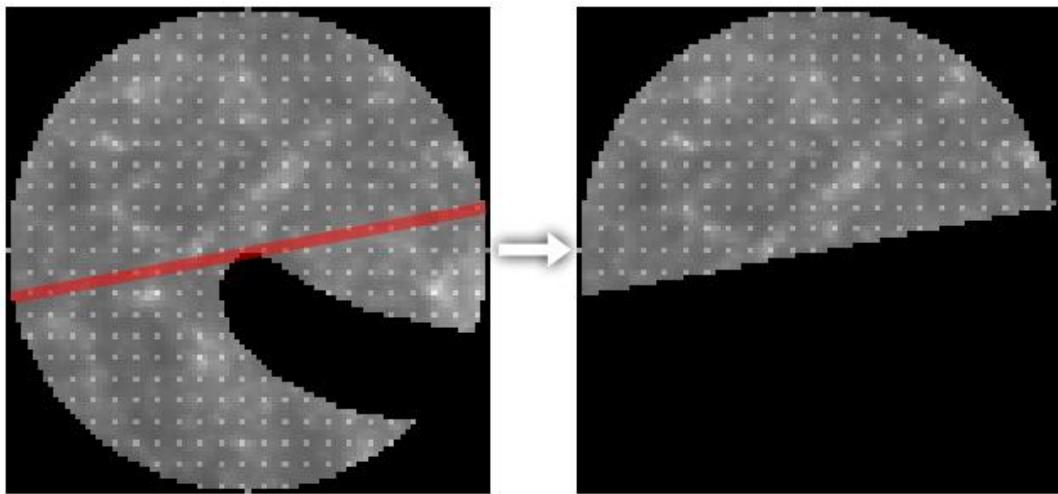
The important thing to note is that the ROI was set with respect to the *current* image (i.e. it uses "handles\_ncorr.set\_roi\_cur(data)"). This is the key point to how Ncorr handles discontinuous displacement fields. The idea is that the discontinuities are clearly visible in the current configuration. The general process is that the ROI and DIC analysis proceed with respect to the current configuration, and then an Eulerian to Lagrangian algorithm converts the displacements back to the reference configuration. The ROI was traced around the crack in Photoshop; a close up of the ROI superimposed on the current image is shown below:



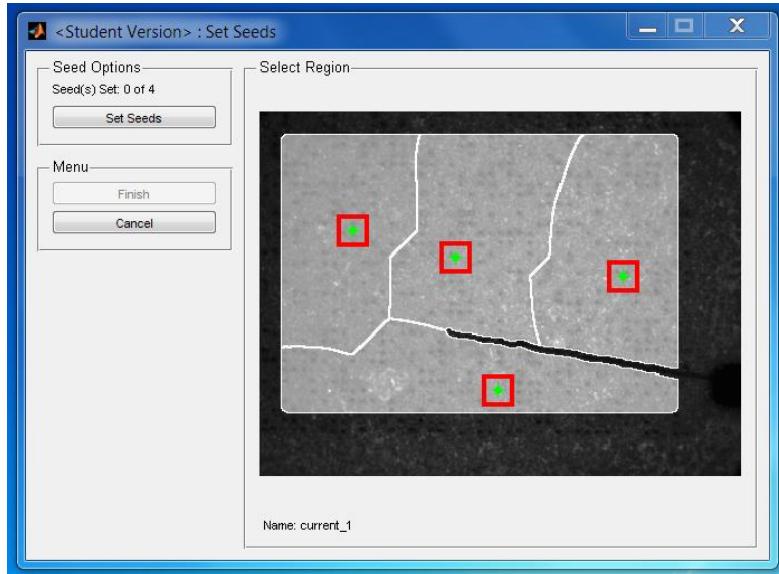
The DIC parameters were then set as shown below:



The one thing to note in particular is that subset truncation is enabled. This prevents wrap around of subsets around the crack tip. This is demonstrated below:

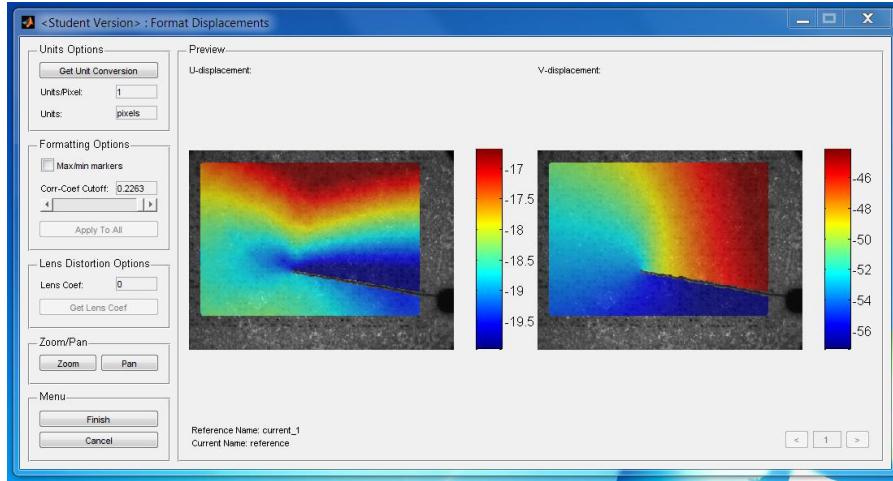


This is useful because the shape functions are linear, whereas the deformation near the crack tip is highly nonlinear. This helps prevent some distortions in the displacement fields which will be shown at the end of this section. Anyway, the seeds are placed as so:

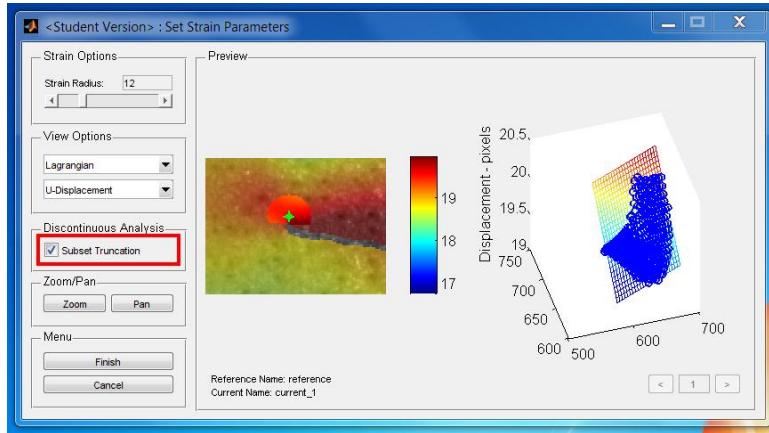


After the analysis is complete, the displacement formatting does not need any special considerations. Just accept the defaults as shown below:

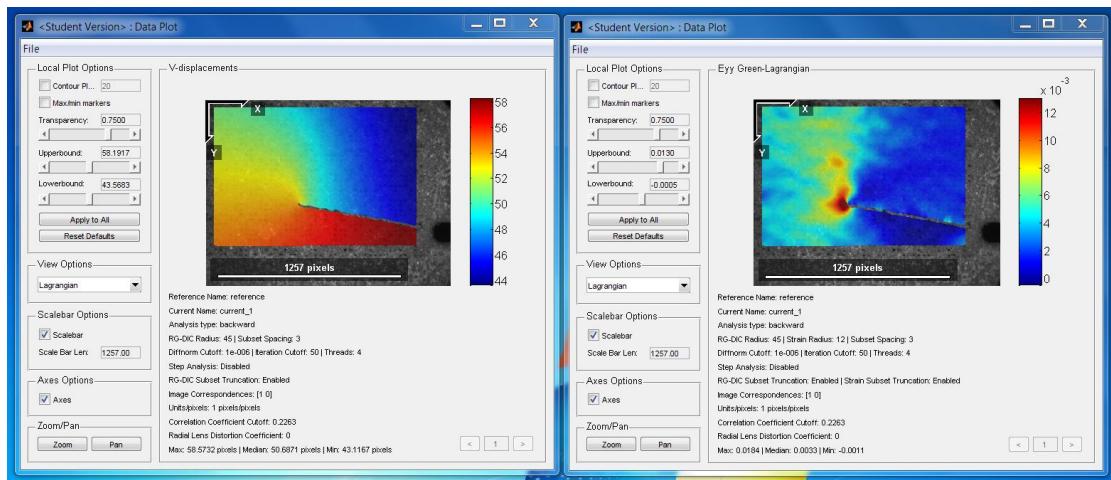
## Discontinuous Displacements DIC Analysis - 4.1



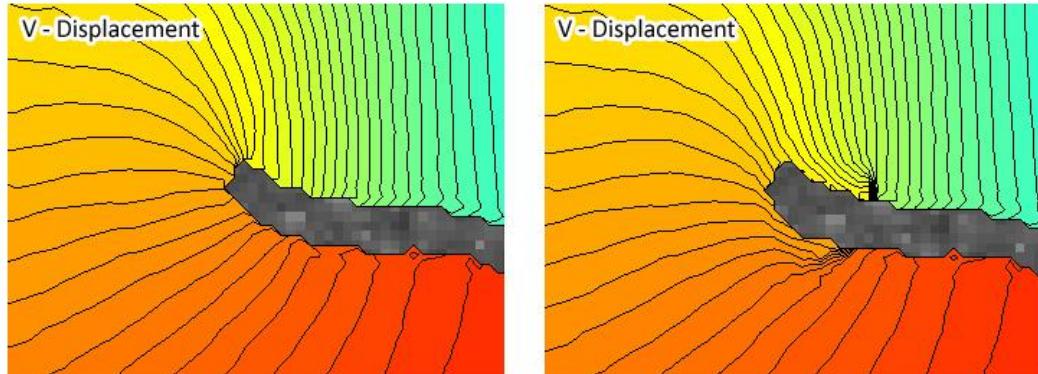
For the strain analysis, make sure to also utilize subset truncation:



After everything is complete, the displacement and strain plots should appear as shown below (only the V displacement and E<sub>yy</sub> strains are shown below since the sample is loaded in the y direction):

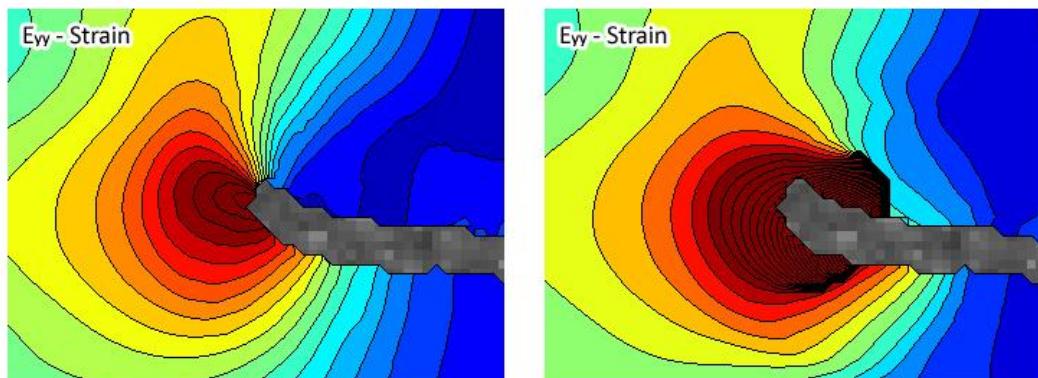


I've taken the liberty of also performing the same analysis without subset truncation. Comparisons are shown below between the zoomed-in V displacement fields (top row) and  $E_{yy}$  Lagrangian strain fields (bottom row) near the crack tip, which demonstrate the benefits of using subset truncation:



With Subset truncation

Without Subset truncation



With Subset truncation

Without Subset truncation